Lifelong Updating of Digital Twin Models for Degraded Systems Using Autoencoder and LSTM

Yifan Tang
Email: yta88@sfu.ca

Mostafa Rahmani Dehaghani
Email: mra91@sfu.ca

G. Gary Wang[*]
Email: gary_wang@sfu.ca

Product Design and Optimization Laboratory, School of Mechatronic Systems Engineering, Simon Fraser University, Surrey, BC, Canada

ABSTRACT

Digital twin (DT) has been applied for monitoring, control, and decision-making of real-world engineering systems, but how to keep a DT valid as the physical system degrades over the lifecycle remains rarely explored. Instead of identifying degradation parameters from expensive aging experiments, this paper proposes a lifelong update method to capture degradation effects directly from system responses via continuously tuning DT models. The core idea is to represent the system degradation as temporal changes of DT model configurations through degradation stages. During the lifelong update process, an autoencoder compresses DT model parameters into latent features, and an LSTM learns the temporal trend of those features to predict latent features for future degradation stages. Based on the predicted latent features, the DT model configuration is reconstructed to predict the system responses affected by degradation. The proposed lifelong update method is evaluated with two real-world engineering datasets, e.g., the battery degradation dataset from Oxford University and the flight engine degradation dataset from NASA. Results demonstrate that the proposed update method successfully captures the degradation effects on system responses during the lifecycle. The one-tailed t-tests confirm that the proposed method statistically outperforms the conventional fine-tuning method in prediction accuracy and robustness at future unseen stages.

Keywords: Digital Twin, System Degradation, Lifelong Update, Battery Degradation, Flight Engine Degradation

[*] Corresponding author.

1 INTRODUCTION

1.1 Literature review

Digital twin (DT) has been adopted widely for health monitoring and process control during products' lifecycle, due to its ability to interact with physical entities (e.g., product, system). Generally, DT is defined as a virtual representation of a physical system [1, 2] that can mimic the time-series response of the physical system under various conditions and is capable of autonomous self-updating based on data acquired from the physical systems by sensing devices. Since the terminology was proposed in 2003, DT applications have been observed in manufacturing [3], process industry [4], aerospace [5], and other engineering tasks [6] to support collaboration, information access, and decision-making.

Before applying DT, historical data (e.g., sensor, simulation) of the physical system is used to construct an offline DT model via modeling methods, such as partial differential equations (PDEs) and data-driven models (e.g., artificial neural networks, support vector machines) [2]. In most applications, historical data is collected from the early stage of its whole lifecycle, during which a practical system continues changing, such as operation conditions, accuracy of sub-systems, and functionality of system components. Therefore, after deploying the DT model, it is necessary to update the DT model with the data stream collected at each stage, so that the DT model would function well throughout the whole lifecycle.

According to the literature review on iterative updates of DT models [7], update methods could be classified according to the model types. For PDEs-based DT models, the update methods are designed to tailor system parameters (e.g., mass or stiffness matrices) that are learned offline. This task is regarded as a parameter estimation problem, which has been solved by Kalman filters [8], Particle filters [9], and Bayesian estimation [10] in data assimilation [11] and hybrid simulation [12], where the mathematical update formulations are derived from system PDEs. For data-driven DT models, the update methods aim to tune model parameters by optimization, Bayesian estimation, or incremental computation [13], or modify the model structure using Bayesian techniques

[14] or heuristic strategies [15]. Although update methods for both model types have been studied, most studies focus on aligning the updated DT models with data collected at the current stage. Therefore, the updated DT model tends to forget system behaviors observed in earlier lifecycle stages and will fail to forecast the system behavior when the physical system changes at future stages.

One important fact is that a physical system degrades continuously in its lifecycle. Such system degradation could be the decreasing battery discharge capacity over charge-discharge cycles [16,17], reduction of the remaining useful life in machines [18,19], and other system health indicators that change over the lifecycle. To monitor a system's health, degradation models have been studied [20]. The first type of degradation model is physics-based, such as the Paris law to predict the crack growth of an operational unit [21,22], the Fick's diffusion law to model the corrosion rate in reinforced concrete [23], the tunnel serviceability index to measure the service state of a tunnel infrastructure [24,25]. However, those physics models require prior knowledge of system physics and massive, expensive experimental data for modeling. The applied assumptions restrict their generalizability in different applications. With the increase in sensor measurements, data-driven models have emerged as the second type of degradation model. For instance, the physics-informed long short-term memory (LSTM) model has been used to predict the degraded battery capacities, which is applied to forecast the battery's remaining useful life [17]. The degraded stack voltage of a fuel cell could be obtained by a hybrid convolutional neural network-LSTM model from the outlet temperature and the historical stack voltage [26]. The dynamic health state of a bearing could be estimated from the sensor signals by an adaptive hidden Markov model [27]. More similar applications could be observed in lithium-ion batteries [28], gas turbines [29], bearings [30]. However, those data-driven models are only health prognostic models using the system responses as input.

The above works directly model system degradation as it happens. In addition, DT has been used to predict system degradations, which generally follows two categories with detailed applications as below.

Category 1: Predict health indicators based on DT predictions of system responses.

The most common application uses the DT model to predict time-series system responses, which are inputted into degradation models to forecast the remaining useful life and other health indicators [31]. For example, based on the incomplete discharge curve and the battery state-of-charge curve at one charge-discharge cycle, a LSTM model was applied to predict the completed discharge curve, and subsequently, the degraded capacity [32]. Similarly, the backpropagation neural network was adopted to predict the whole discharge curve [16]. The battery degradation state was then estimated by a convolutional neural network-LSTM-attention model from the predicted discharge curve and the historical discharge curves of previous cycles. For the proton exchange membrane fuel cell stack, the transformer was adopted to predict the future stack voltage based on time-series operation parameters (e.g., temperature, current) and the observed voltage [33]. The remaining useful life is then estimated as the time when the future voltage is smaller than a threshold.

When obtaining data at one degradation stage, those models could be fine-tuned to provide accurate prediction at the current stage. However, system responses/signals at future degradation stages are unknown for practical applications, making them unable to forecast the degraded system responses for future stages in advance.

Category 2: Construct DT models with identified degradation parameters of physical systems.

Instead of updating the entire DT model, some works construct DT models with identified degradation parameters, which are learned at each degradation stage. For instance, Simulink-based DT models were proposed based on identified degradation characteristics parameters (e.g., capacitance, inductance, parasitic resistance) for DC/DC power converters [34] and the DC/AC inverter [35]. At each degradation stage, those parameters are updated with actual data via Bayesian optimization or particle swarm optimization. Although both works can make the updated Simulink-based DT model capture the actual system behavior at each degradation stage, they cannot forecast the degraded system responses at future stages in advance, as the identified parameters continue degrading over the lifecycle.

To avoid the above limitation, degradation models for degradation parameters are trained based on historical data to capture the trend of degradation parameters over the lifecycle, enabling more practical DT predictions for future stages. For instance, a high-fidelity finite element model was developed for the electronically controlled pneumatic brake system, where the key degradation parameter is the direct-current resistance [36]. A data-driven model was trained to capture the resistance degradation trend, allowing reinitialization of the model at future stages. Similarly, for the gearbox, a high-fidelity dynamic DT model was proposed with two degradation parameters, the pitting density and the wear depth [37]. Fatigue pitting and Archard wear models were trained on historical data to capture their propagation over time [38]. During deployment, DT predictions were compared with physical system responses to iteratively update the degradation models, which in turn provided more realistic degraded parameters to refine the initial geometry setting in the dynamic DT model for better prediction.

Apart from updating degradation models during applications, some studies pretrain an offline degradation model and apply the model without any further update. For instance, to predict the real-time temperature of lithium-ion batteries, a lumped thermal equivalent circuit (i.e., DT model) was designed with the heat capacity and the thermal generation rate [39]. The degradation of both parameters was captured by an LSTM model from the operation history data so that the DT model could update its model setting at each degradation stage for better prediction. Similarly, considering the thickness of the oxidation layer grows in semiconductor devices, a finite element model was designed based on affected mechanical properties, e.g., geometry and material behavior [40]. The experiments were conducted to model those properties as functions of aging, which makes the finite element analysis successful at each degradation stage.

Although the above DT models could capture degradation to some extent, they share two fundamental limitations. First, the DT models in both categories rely on system responses affected by degradation at the current stage. Those responses can improve the prediction performance of DT models at the current stage, but they cannot forecast the degraded system responses at future stages before the data are observed. Second, the update methods in Category 2 require a prior knowledge of physical degradation parameters (e.g., resistance) and typically need expensive, long-term aging experiments to capture their degradation behavior. For complex engineering systems, such degradation parameters may be unknown or not explicitly measurable, and collecting sufficient degradation data is often infeasible in practice.

Above discussions show that existing update methods either fail to support the model to generalize to future unknown degradation stages or assume the degradation parameter is known and modeled in advance. Therefore, the fundamental gap to deploy DT models is that there is no DT update method, which can represent system degradation implicitly without prior knowledge of degradation parameters, and enable the prediction in advance of degraded system responses at future stages.

1.2 Motivation

Rather than introducing modeling approaches, this paper aims to address the fundamental gap, i.e., *how to keep existing DT valid through the lifecycle via capturing degradation implicitly to support prediction of future degraded performance*. Given a DT model and its corresponding physical system, this paper proposes a lifelong DT update method to ensure accurate predictions at future degradation stages. When updated with online data from one degradation stage, the DT model memorizes degradation effects on the system responses at earlier stages, based on which system responses affected by progressive degradations are predicted. This method differs from existing approaches in two ways. (1) Different from current online DT update methods, the proposed method supports prediction of future degraded time-series responses without prior knowledge of future degradation in advance. (2) Compared with reviewed DT applications in system degradation, the proposed method is cost-effective as it avoids the need for explicitly identified degradation parameters, expert knowledge, or expensive offline experiments, which would improve its applicability in different applications.

To address the aforementioned gap, this paper makes the following contributions.

- A novel DT update task is defined for systems degrading during their lifecycle, aiming to predict future system degradation without prior knowledge or costly experiments. According to the authors' best knowledge, this DT update task was rarely studied in existing literature.

- A lifelong DT update method is proposed for DT models constructed as feedforward neural networks. Instead of updating DT models stage by stage, the method learns degradation patterns from DT model parameters and predicts DT configurations at future, unseen degradation stages in the lifecycle.
- The method offers an economical alternative to traditional aging tests, while enabling predictions of future system degradation effects, which can support system health management, not provided by conventional fine-tuning methods.

The remainder of the paper is structured as follows. The representation of system degradation in the DT model is discussed in Section 2.1, based on which the lifelong DT update method is discussed in detail in Sections 2.2 and 2.3. The proposed method is then tested with the battery degradation dataset in Section 3 and the flight engine dataset in Section 4. Discussions are presented in Section 5, followed by a summary of the work in Section 6.

2 METHODS

2.1 Representation of system degradation in DT

Degradation in physical systems reflects a continuous and gradual system performance deterioration over the lifecycle [41]. Such continuous performance change indicates that even under the same operation setting, time-series responses of the same physical system differ among degradation stages in the whole lifecycle. Capturing these dynamics necessitates an approach that allows for the seamless representation of degradation effects on the system responses in the lifecycle.

In this paper, DT models are constructed only based on simulation or experimental data via machine learning models, such as neural networks. In the conventional fine-tuning framework to update the data-driven DT model throughout the system lifecycle, the "optimal" DT model is iteratively tuned for each degradation stage when the new dataset is obtained. Specifically, a DT model with configuration $\boldsymbol{\theta}_0$ is trained using data collected at the initial degradation stage T_0 (e.g., no system degradation). As the system degrades over time, new data from subsequent degradation states (e.g., T_1) are used to update the DT model, resulting in a new configuration $\boldsymbol{\theta}_1$ that captures system behavior at the stage T_1. This update process is iteratively applied at each degradation stage. Therefore, the DT configuration $\boldsymbol{\theta}_i$ at the degradation stage T_i is regarded as a representation of the degradation effect on system responses at the stage T_i in this study. Compared with reviewed works relying on degradation parameters, the selected degradation representation only depends on DT model configuration, offering a more general and adaptable solution for a wide range of physical systems.

2.2 Lifelong DT update method

As mentioned in Section 1.2, this paper aims to propose a lifelong DT update method, so that the updated DT model is able to predict the time-series system's future responses directly affected by system degradation. To fulfill this purpose, a model that captures the dynamic system degradation over the lifecycle is required, such as the reviewed works using the degradation parameters or a degradation model. Instead of using prior knowledge, constructing a dynamic model of DT configurations over degradation stages would be a promising counterpart, according to the defined degradation representation. More specifically, if a dynamic model $\boldsymbol{\theta}_i = f(\cdot, T_i)$ is learned during the lifelong update process, DT configurations at future degradation stages could be predicted directly, thereby capturing the system responses affected by degradation at future stages.

By integrating the tuning process and the dynamic model construction, the lifelong DT model update method is proposed as shown in Fig. 1, whose detailed steps are described below. The corresponding pseudo codes are presented in Algorithm 1 at the Appendix.

Step 1 (Initialization phase): Given a physical system, the initial data-driven DT model DT_0 is constructed on a dataset $\boldsymbol{D}_0$ obtained at the stage T_0, where no degradation is assumed. The obtained DT configuration $\boldsymbol{\theta}_0$ is stored in the configuration database, based on which the dynamic model structure is determined and remains fixed during the future lifelong update process.

Step 2 (Warm-up phase): In most real-life scenarios, the physical system would work well during the early stage of the lifecycle. In this paper, a group of m earlier degradation stages $\{T_1, \dots, T_m\}$ is regarded as the warm-up phase, where the DT model is continuously fine-tuned based on the system response dataset collected at each degradation stage. Configurations of the updated DT model at each stage are saved in the configuration database. Based on the stored configurations $\{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_m\}$, the dynamic model $\boldsymbol{\theta}_i = f(\cdot, T_i)$ is trained, whose details are presented in Section 2.3.

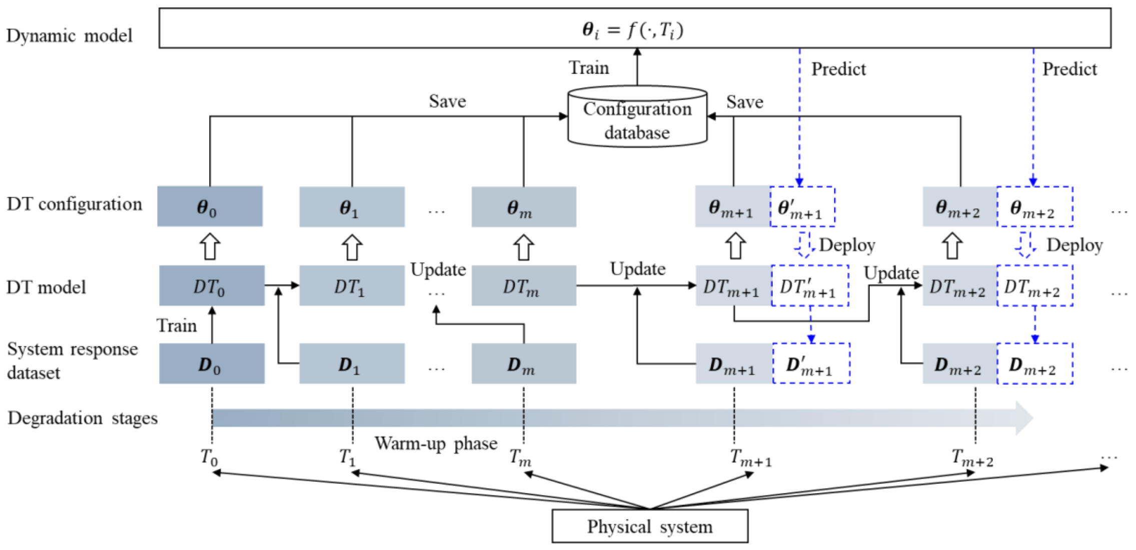


Fig. 1. Proposed lifelong DT update method. The solid lines and arrows indicate the lifelong update process, and the dashed lines and arrows refer to the application of the DT.

Step 3 (Lifelong application phase): For any future degradation stage T_i $(i > m)$, the trained dynamic model f is adopted to predict the DT configuration $\boldsymbol{\theta}'_i$, based on which the estimated DT model DT'_i is obtained. The system responses affected by degradation at future stages $\{T_i, \dots, T_{i+j}\}$ are then predicted by the corresponding estimated DT models $\{DT'_i, \dots, DT'_{i+j}\}$. The estimated system responses $\{\boldsymbol{D}'_i, \dots, \boldsymbol{D}'_{i+j}\}$ could be applied to calculate the health state of the system for management and other services, which will not be covered in the paper.

Step 4 (Lifelong update phase): After completing the task at the i-th degradation stage T_i, the actual dataset $\boldsymbol{D}_i$ is obtained to update the DT model DT_{i-1} with configuration $\boldsymbol{\theta}_{i-1}$. The updated model is denoted as DT_i, whose configuration is $\boldsymbol{\theta}_i$. Compared with the estimated configuration $\boldsymbol{\theta}_i{}'$ from the previous dynamic model, the new configuration $\boldsymbol{\theta}_i$ captures the actual effect of degradation on system responses at the stage T_i. To improve the accuracy of the proposed dynamic model gradually, the new configuration $\boldsymbol{\theta}_i$ is stored in the configuration database to retrain the dynamic model. More details are discussed in Section 2.3.

2.3 Dynamic model for DT configuration

Generally, the configuration of a data-driven DT model includes the structure and parameters, which could be updated simultaneously, such as the works in [14, 15]. In this paper, a feedforward neural network (FNN) with a fixed structure is adopted as the DT model, meaning that the dynamic model only needs to capture how the DT model parameters change during the whole lifecycle.

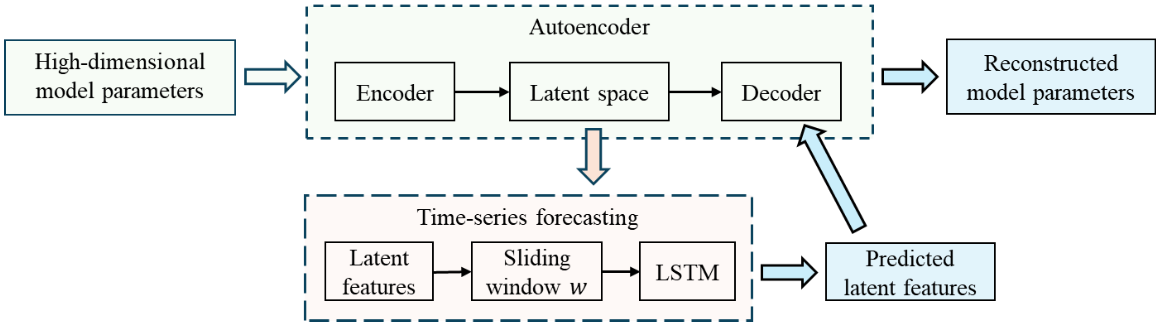


Fig. 2. Conceptual framework of the proposed dynamic model.

The general framework of the proposed dynamic model is summarized in Fig. 2, where Autoencoder is adopted to capture the latent features of DT configurations at each degradation stage, and LSTM is used for time-series forecasting of future configuration features based on features at the previous w degradation stages. In this work, the latent features serve as a lower-dimensional representation that encodes the inherent patterns in the high-dimensional DT model parameters. Since these model parameters are fine-tuned to reflect the degradation effects on system responses at each stage, the latent features can be interpreted as a compact representation of the underlying physical degradation mechanisms, to some extent. In the following discussion, the proposed dynamic model is denoted as Autoencoder-LSTM (AE-LSTM) for clarification.

2.3.1 Autoencoder to reconstruct model configurations

Considering a FNN model with N hidden layers $\{l_1, \dots, l_N\}$, the input dimension and the output dimension of the i-th hidden layer l_i are n_{in}^i and n_{out}^i respectively. Each hidden layer uses a linear transformation, e.g., $y = xA^T + b$, where A is the weight matrix and b is the bias vector. Therefore, the number of parameters at the hidden layer l_i is $\left(n_{in}^i + 1\right) \times n_{out}^i$. The total number of parameters in the model is $\Sigma_{i=1}^N \left(n_{in}^i + 1\right) \times n_{out}^i$. This number could be thousands or millions, making the task of capturing the dynamic behavior of model configuration a high-dimensional problem. If each parameter is learned by a separate model, a large memory is required to store thousands of models. If a simple FNN is used to predict the configuration based on the degradation stage, the output dimension is then huge, making the training process harder to converge. Both options are thus not ideal and may risk poor prediction performance at future degradation stages.

To reduce the problem dimension, an autoencoder is utilized to reconstruct the model configuration $\boldsymbol{\theta}$. The encoder learns efficient low-dimensional features, based on which the decoder reconstructs the entire model configuration. Autoencoder has been a popular tool for feature extraction in deep learning. Compared to traditional dimension reduction techniques (e.g., proper orthogonal decomposition), the autoencoder can capture complex nonlinear relationships. Unlike supervised learning approaches, Autoencoder does not require labeled data, which is advantageous when dealing with large-scale model configurations where annotations are unavailable. Based on those merits, Autoencoder has been applied in fluid dynamics [42], material [43], and other engineering fields [44].

Fig. 3 shows the proposed autoencoder structure. Given the model configuration $\boldsymbol{\theta}$, the weights and biases of each hidden layer l_i are flattened as a parameter vector with the size $1 \times \left(\left(n_{in}^i + 1\right) \times n_{out}^i\right)$. The encoder takes parameter vectors of all hidden layers as inputs simultaneously, aiming to capture the inherent relationship among all hidden layers. To alleviate the effects of various dimensions, each parameter vector is compressed to a vector with the same size 1×64 by the corresponding linear hidden layer. The compressed vectors of N hidden layers are then concatenated into one vector with the size $1 \times 64N$, which are processed by three linear hidden layers with the activation function Tanh(·). The output is then process by another linear hidden layer to obtain the latent space presentation. In the latent space, L is the dimension of configuration features extracted from the input model configuration. Once the configuration features are obtained, they are fed to the decoder, where four hidden layers with the activation function Tanh($\cdot$) are adopted to output the concatenated vector with the size $1 \times 64N$. The vector is then divided to N separate vectors with the same dimension 1×64. Each split vector is processed by a linear hidden layer to reconstruct the parameter vector in the DT model. The reconstructed parameter vectors are then deployed to the DT model for system response prediction.

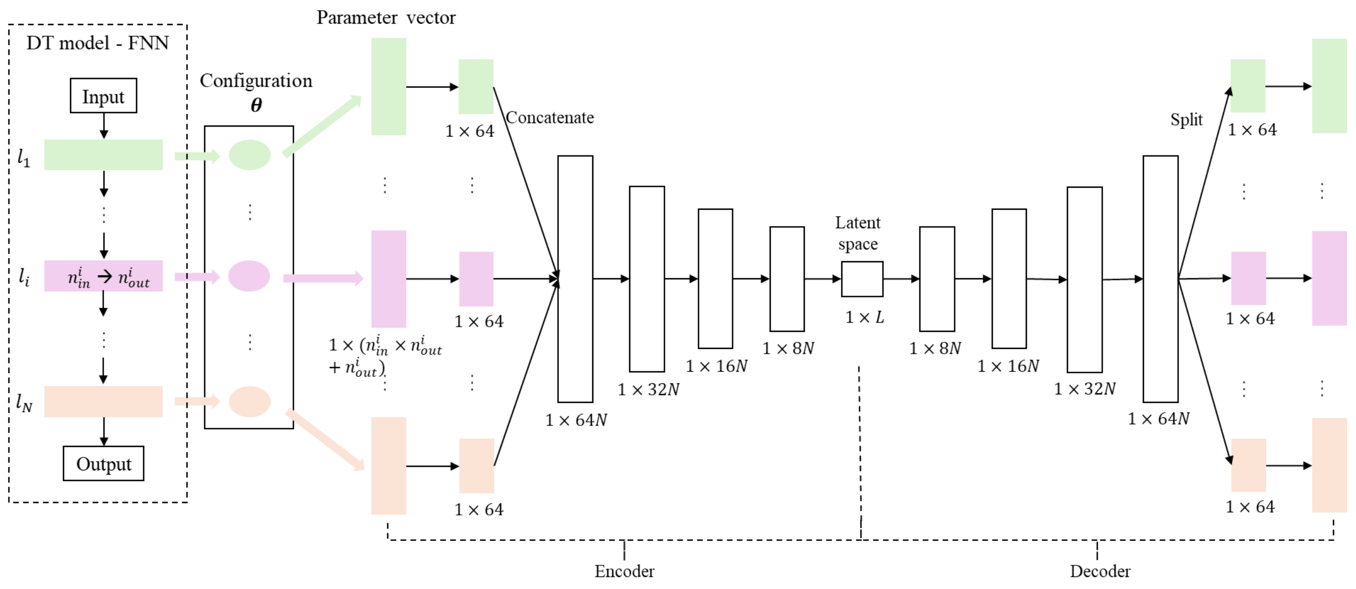


Fig. 3. Autoencoder structure to reconstruct the high-dimensional configurations.

When designing an autoencoder, selecting an appropriate dimension L of the latent features is fundamental, as it affects the model performance in terms of data reconstruction, feature extraction, and generalization [45]. If the dimension is too low, the autoencoder may not capture essential features, resulting in a loss of information. Conversely, an excessively large latent space would cause overfitting and fail to capture the underlying data structure. The most applied method involves training an autoencoder with varying latent dimensions and evaluating their impact on downstream tasks (e.g., classification accuracy) and reconstruction error, which is computationally intensive [46]. The dimension of latent features could also be determined by the singular value decomposition method, where dimensions of singular values above a threshold are retained [47], or by incorporating the regularization term of latent space during the training process to encourage the encoder utilizing fewer latent features [48]. However, the performance of both methods depends on the dataset, model structure, and downstream task.

Instead of finding the optimal dimension, this paper determines the dimension L of latent features based on the information theory and the model configuration $\boldsymbol{\theta}_0$ at the initial degradation stage T_0. The dimension is then fixed during the lifelong update process. In data compression tasks, the Shannon entropy is regarded as the minimum number of bits to compress a group of random variables [49,50]. According to [51], the stochastic behavior of model parameters during the training process could be considered as a Gibbs distribution, based on which three equations are designed to determine the latent feature dimension automatically according to the given initial DT model. Generally, Equation (1) converts each parameter into a probability based on its magnitude. Equation (2) measured the Shannon entropy of the converted probability distribution, where a higher entropy means the model parameters contain more information. Equation (3) finally sets the latent dimensions proportional to the entropy value, so that the latent space would have sufficient capacity to capture the information of DT models at future degradation stages. More details are presented as bellow.

Assuming that each model parameter could be regarded as sampled randomly from the same distribution, the probability of each model parameter could be estimated as:

$$p_{l_i}^j = \frac{\exp\left(-\beta\theta_{0,l_i}^j\right)}{\sum_i^N \sum_j^{\left(n_{in}^i+1\right)\times n_{out}^i} \exp\left(-\beta\theta_{0,l_i}^j\right)} \tag{1}$$

where θ_{0,l_i}^j is the j-th model parameter of the i-th hidden layer l_i at the degradation stage T_0, $p_{l_i}^j$ is the corresponding probability, and β is a constant value. The Shannon entropy of the model configuration $\boldsymbol{\theta}_0 = \{\theta_{0,l_i}^j | i \in [1, N], j \in [1, \left(n_{in}^i + 1\right) \times n_{out}^i]\}$ is then calculated as Eq. (2).

$$H = -\sum_{i,j} p_{l_i}^j \log_2 p_{l_i}^j \tag{2}$$

The dimension L is finally determined as:

$$L = a \times [H] \tag{3}$$

where $[H]$ is the integer number closest to the entropy H. Considering that the model configuration evolves continuously during the system lifecycle, the entropy of model configurations varies at different degradation stages. Therefore, $a > 1$ is a ratio to enlarge the integer value, which would reduce the risk of the low-dimensional latent space failing to capture the underlying features at future degradation stages. In the paper, the hyperparameters are set as $\beta = 1$ and $a = 2$ when testing the two engineering datasets.

2.3.2 LSTM to predict configuration features

After training the autoencoder, the learned latent features $\boldsymbol{s}_i$ of model configurations at all known degradation stages are accessible. When using the learned autoencoder to predict configurations, the autoencoder structure is fixed and the prediction performance only depends on the estimated latent features. As discussed in Section 2.1, the system degradation is represented as the dynamic behavior of latent features over the lifecycle. Therefore, a time-series forecasting model is required to estimate the latent features at future degradation stages.

In time-series forecasting tasks, LSTM has been a powerful tool due to its ability to capture long-term dependencies in sequential data while mitigating the vanishing gradient problem [52]. Unlike traditional recurrent neural networks, LSTM incorporates gating mechanisms, i.e., input, forget, and output gates, which selectively retain relevant information and discard irrelevant information. This characteristic makes LSTM effective in capturing complex temporal patterns over sequential data, such as financial market trends, weather prediction, and industrial processes. Furthermore, LSTM outperforms classical time-series models by handling non-linearity and non-stationary time-series data without requiring extensive feature engineering. Based on these advantages, LSTM has been observed in a wide range of applications, including financial market forecasting [53], healthcare analytics [54], and anomaly detection [55].

Based on the above discussion, this paper adopts the LSTM as the time-series forecasting model for latent features of model configurations. In theory, an optimal time-series forecasting model exists for the proposed task, where the number of hidden layers, dimension of hidden states in the LSTM cell, activation function, and other hyperparameters are optimized by hyperparameter optimization [56]. As hyperparameter optimization is not the research topic of this work, the proposed model structure is determined by trial and error, instead of the expensive optimization.

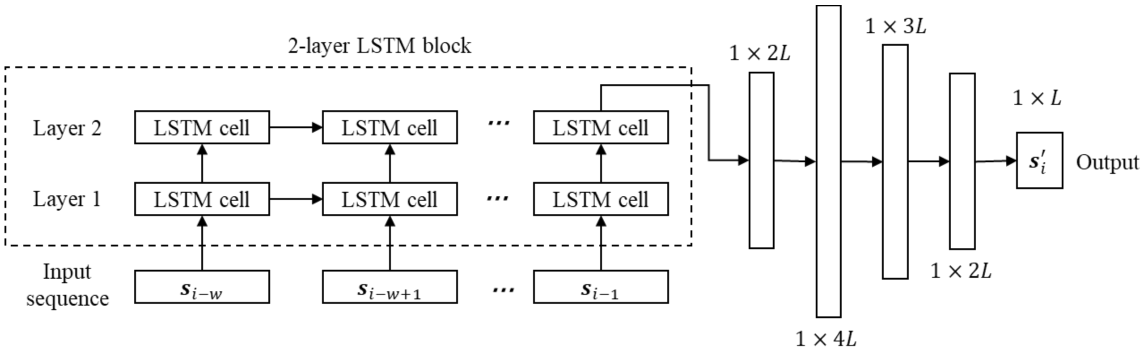


Fig. 4. LSTM structure to predict the configuration feature.

Fig. 4 summarizes the model structure. In the model, a two-layer LSTM block is adopted. The input sequence $\boldsymbol{s}_{i-w}, \dots, \boldsymbol{s}_{i-1}$ with each element having the dimension of $1 \times L$ is processed by the first LSTM layer, whose output hidden states are input to the second LSTM layer. In this work, the LSTM block is implemented with the hidden state size $1 \times 2L$ and two LSTM layers using Pytorch, and the dropout rate is zero. The output state of the last cell in the second LSTM layer is then processed by several linear hidden layers with the activation function Tanh(·) to obtain the final output as the predicted latent feature vector $\boldsymbol{s}_i'$ with the size $1 \times L$.

Once the proposed LSTM model has been trained, a progressive prediction framework is adopted to predict latent features of model configurations at all future unknown degradation stages. As shown in Fig. 5, if we know all latent features before the i-th degradation stage and want to predict all latent features at each future stage, an input matrix $[\boldsymbol{s}_{i-w+1}; \dots; \boldsymbol{s}_i]$ with the window size w is fed to the trained LSTM model to predict the latent feature $\boldsymbol{s}_{i+1}'$ at the $(i+1)$-th degradation stage. Then the updated input matrix $[\boldsymbol{s}_{i-w+2}; \dots; \boldsymbol{s}_i; \boldsymbol{s}_{i+1}']$ is adopted to obtain the estimated latent feature $\boldsymbol{s}'_{i+2}$ at the $(i+2)$-th degradation stage. This process will repeat until the latent feature at each future degradation stage is obtained. Based on those predicted latent features, the estimated model configuration at each future degradation stage is obtained by the decoder in the trained autoencoder model, which would be applied to predict the system responses affected by the system degradation.

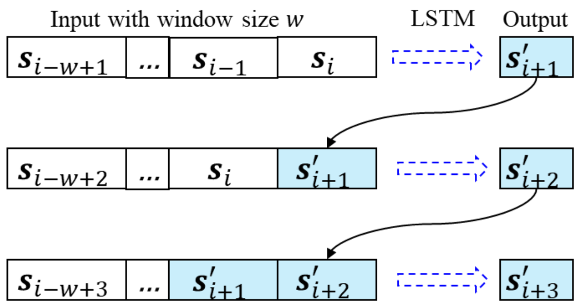


Fig. 5. Prediction of future configuration features.

3 OXFORD BATTERY DEGRADATION DATASET

3.1 Dataset description and preparation

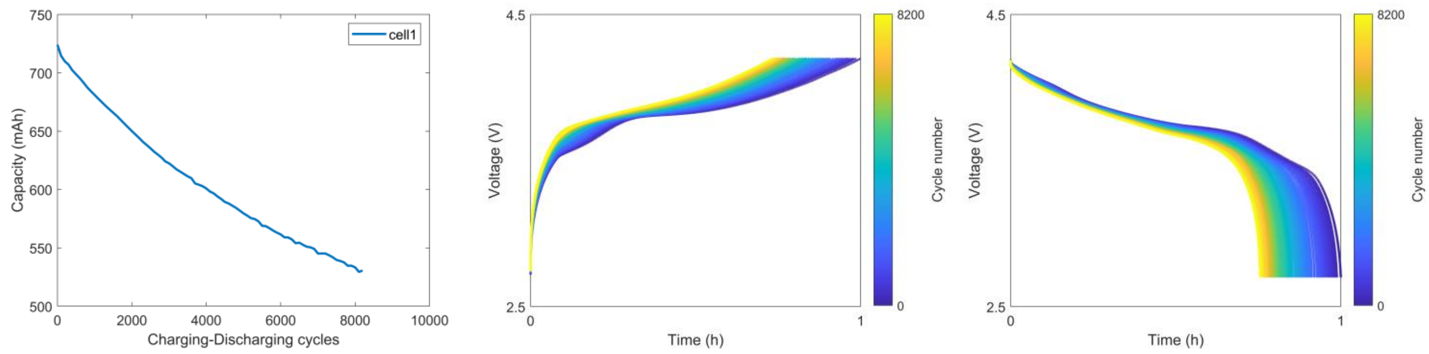
The battery dataset is collected by the Battery Intelligence Lab at Oxford University [57]. The dataset contains measurements from the eight SLPB533459H4 batteries produced by Kokam CO LTD, which are tested by Bio-Logic MPG-205 with eight channels. During the test, the battery works under a constant-current-constant-voltage charging profile, followed by a drive cycle discharging profile. The charging-discharging rate is 1C, which means the battery is charged/discharged at a current equal to its rated capacity. The specific basic parameters of the battery are summarized in Table 1. Among the eight cells in the battery, only the first cell is studied in the test.

In the dataset, the battery is charged and discharged repeatedly until the battery capacity reduces to a specified value when the battery meets the retirement requirement. As the service life of the battery is quite long, the battery properties (e.g., time, voltage, charge, and temperature) are measured and stored every 100 charging-discharging cycles. The degradation stages in Section 2.2 are then defined accordingly, e.g., regarding Cycle 0 as the stage T_0, the cycle 100 as the stage T_1, and the cycle 8200 as the stage T_{82}. Based on the stored measurements, Fig. 6 summarizes degradation curves in Cell 1 of the battery over charging-discharging cycles (i.e., the lifecycle).

According to Fig. 6 (a), the battery capacity obviously decreases with the charging-discharging cycle number, indicating that the battery health degrades continuously during the lifecycle. Affected by the battery capacity degradation, the time to charge the battery from 2.7V to 4.2V and the time to discharge the battery from 4.2V to 2.7V reduce with the cycle number, as shown in Fig. 6 (b) and Fig. 6 (c), respectively.

Table 1. Parameter of the tested battery in the Oxford battery degradation dataset (Revised from [16])

Parameter	Value	Unit
Constant current	740	mA
Rated capacity	740	mAh
Voltage range	2.7-4.2	V
Environmental temperature	40	℃
Cell number	8	



(a) Discharge capacity over cycles (b) Voltage curves in charging (c) Voltage curves in discharging

Fig. 6. Battery degradation curves.

In this test, the proposed lifelong update method aims to capture the effect of the degraded capacity on the voltage curves and enable the DT model to forecast the affected voltage curves at all future unseen charging-discharging cycles. More specifically, two FNN models are used as DT models for the charging voltage curve and the discharging voltage curve. According to the defined task in Section 1.2, the partial voltage curves at future charging-discharging cycles are unknown in advance in this work. This is different from works reconstructing the whole voltage curve based on a partial curve, which only works at the known stages [32,33]. Therefore, the inputs of both FNN models are defined only based on the predefined charging or discharging settings, i.e., constant current $c = 0.74A$, and time t (s). The output is the voltage v at the time t when charging or discharging with a constant current. Considering that the number of data points on each curve is thousands at each cycle in the dataset, 200 data points, i.e., voltages with corresponding time steps, are sampled evenly from the raw voltage curves to simplify the modeling task. In other words, the charging and discharging curves at each cycle contain 200 data points respectively. Finally, datasets covering 80 cycles (200 data points on each cycle) are obtained for both charging and discharging curves, as the data from cycles 3400, 4700, and 4900 (i.e., degradation stages 34, 47, and 49) are not given in the published dataset.

3.2 Test setting

The proposed lifelong update process consists of several training components, including the training of initial DT models DT_0, the fine-tuning of DT models based on data from each degradation stage, and the training of the dynamic model for the model configuration during the lifecycle. During this test, the DT models for the charging and discharging curves share an identical structure, as shown in Fig. 7. As the nonlinearity in both charging and discharging curves is low, a simple structure using four hidden layers is utilized to predict the voltage from the constant current and time. All hidden layers use the same activation function Tanh(·). Based on 200 data points for each of the charging and discharging curves at the initial degradation stage T_0, the model DT_0 is trained with the detailed hyperparameters in Table 2. The epoch number is 1,000. The loss function is the mean square error (MSE) between the predicted and actual outputs, which is minimized by the Adam optimizer with the learning rate 0.005.

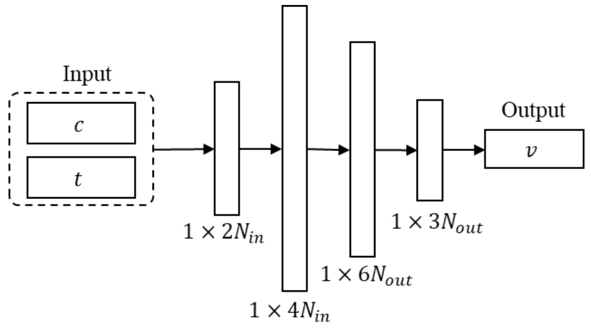


Fig. 7. FNN structure for battery dataset. $N_{in} = 2$ and $N_{out} = 1$ are the input and output dimensions of the model, respectively. The model input contains a constant current c and time t. The output is the voltage v.

Once the initial DT model is trained, the DT model is fine-tuned sequentially in the warm-up phase, which consists of $m = 20$ degradation stages, i.e., $\{T_1, \dots, T_{20}\}$. At each stage, all collected data is used to update the DT model, with the epoch number 10, the learning rate 0.001, and the optimizer Adam. Different from training the initial DT model, the loss function for the fine-tuning process is defined as below,

$$loss = MSE + \alpha \times l^2(\cdot) \tag{4}$$

where $l^2(\cdot)$ is the 2-norm regularization term on model parameters, so that the model parameters would not change drastically after tuning, and the overfitting risk is reduced. α is the coefficient for the regularization term, which is set as 10^{-5} in the test.

Table 2. Training settings when testing the lifelong update method with the battery dataset.

	Hyperparameter	Value
	Number of stages in warm-up phase, m	20
	Ratio to enlarge entropy of model parameters, a	2
	Constant value in Gibbs distribution, β	1
	Window size for input sequence of LSTM, w	5
Train DT_0	Epoch number	10^3
	Learning rate	5×10^{-3}
	Loss function	MSE
	Optimizer	Adam
Fine-tune DT models	Epoch number	10
	Learning rate	10^{-3}
	Loss function	MSE, l^2-norm regularization
	Weight of l^2-norm in the final loss, α	10^{-5}
	Optimizer	Adam
Train Autoencoder and LSTM	Epoch number	10^3
	Learning rate	10^{-4}
	Loss function	MSE, l^2-norm regularization
	Weight of l^2-norm in the final loss, α	10^{-4}
	Optimizer	Adam

In both charging and discharging, the entropy values of $\boldsymbol{\theta}_0$ are around 6.9, and the dimensions of the latent space in autoencoders are calculated as 14 during the test. When training the proposed autoencoder based on the stored model parameters, the epoch number is set to 1,000, and the learning rate is 0.0001. The loss function is the same as Eq. (4) to maintain the parameter stability in the deep structure, where the coefficient for the regularization term is 10^{-4} in the test. The optimizer is still Adam. Once the latent features of model parameters are captured from the trained autoencoder, the LSTM is trained with the same training setting as the autoencoder.

According to the authors' best knowledge, the online update task proposed in this work, where the effect of system degradation is modelled in a DT update process, has not been discussed in the current literature review. To demonstrate the effectiveness of the proposed lifelong update method, the conventional fine-tuning framework is adopted for comparison. The hyperparameter setting for the conventional fine-tuning framework is the same as the one for fine-tuning DT models in the proposed lifelong update method.

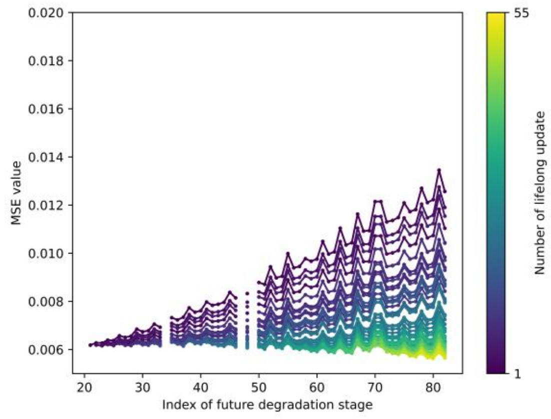
3.3 Results

Based on the above processed datasets from 80 degradation stages and the defined DT model structure, the proposed method is evaluated. According to the description in Section 2.2, when the actual data from the i-th degradation stage T_i is obtained during the lifelong update phase, the previous DT model DT_{i-1} is fine-tuned to get the new model DT_i. The new DT configuration $\boldsymbol{\theta}_i$ is stored with all previous configurations to retrain the AE-LSTM model, which is used to predict the DT model $DT_j'|j > i$ at any future j-th future degradation stage. In such cases, two types of predictions could be obtained for the j-th stage, i.e., the one obtained from the model DT_i, and the one obtained from the model DT_j'. Compared with the actual data at the j-th stage, the MSE value calculated from the former prediction indicates the performance of the fine-tuned DT model DT_i, while the MSE value calculated from the latter prediction reflects the performance of the degraded model DT_j' generated from the proposed lifelong update method. Therefore, after completing the update process at the i-th stage, MSE values at all future degradation stages could be obtained for the fine-tuned DT model and the AE-LSTM generated DT. To clarify the following discussion, the AE-LSTM generated DT at the i-th stage is defined as a group of estimated DT models $\{DT_j'|j = i + 1, i + 2, ...\}$ after completing the proposed updating process at the same stage.

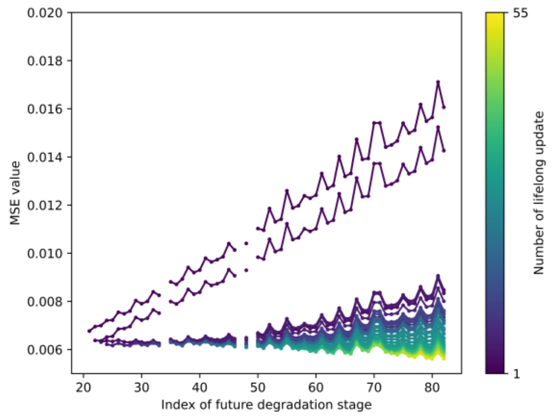
During the test, the last five degradation stages are not used for updating process, so that the DT performance could be evaluated after the last update. Fig. 8 summarizes MSE values at all future degradation stages obtained by fine-tuned and AE-LSTM generated DT models after each lifelong update. For example, after the first update at the 20th stage, all MSE values at future stages obtained by AE-LSTM generated DT are shown as the dark blue curve starting from the 21st stage in Fig. 8 (b). The MSE values obtained after the last lifelong update are plotted as the lowest yellow curves. In the figure, the gap between two successive MSE curves reflects the performance improvement of the DT model after each lifelong update. Compared with the MSE value curves obtained by the fine-tuned DT, the AE-LSTM generated DT demonstrates greater performance improvement after several lifelong updates, as indicated by the large gaps between earlier MSE curves in Fig. 8 (b) and (d). However, it may provide poorer MSE performance during early update steps. This is attributed to an insufficient predefined warm-up phase, failing to provide sufficient information of degradation behavior, which hinders AE-LSTM effectively capturing the degradation effects. Despite this, after a certain number of online updates, the overall range of MSE values at each future degradation stage is smaller in the AE-LSTM generated DT in most cases, which is discussed in detail below.

To clarify the performance difference at each degradation stage between fine-tuned and AE-LSTM generated DT models, Fig. 9 summarizes the box plots of MSE values at each future stage during the whole lifelong update phase, except for the first 10 update steps. For instance, the two MSE values at the 32nd stage are collected from each DT after completing the 11th lifelong update step at the 30th stage and the 12th lifelong update step at the 31st stage. The number of MSE values at i-th stage in the figure could be calculated as $i - 10 - m$, where m is the number of stages in the warm-up phase. The figures indicate that for both battery charging and discharging datasets, the AE-LSTM generated DT outperforms the fine-tuned DT models at most future degradation stages, in terms of smaller median MSE values (i.e., the orange line in the box) and smaller box sizes. The comparison results demonstrate that during the lifelong update phase, the proposed method can capture the effect of system degradation on the system responses at future unseen degradation stages in advance, better than the fine-tuned

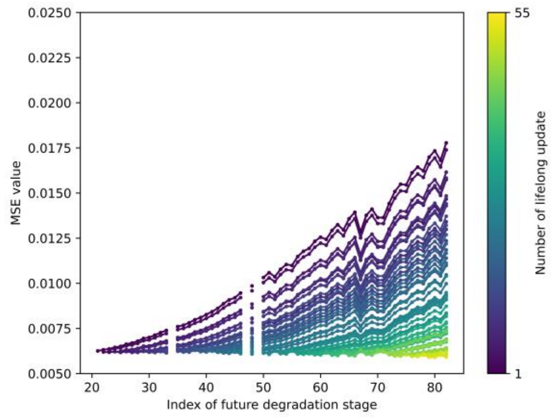
DT with smaller variations.



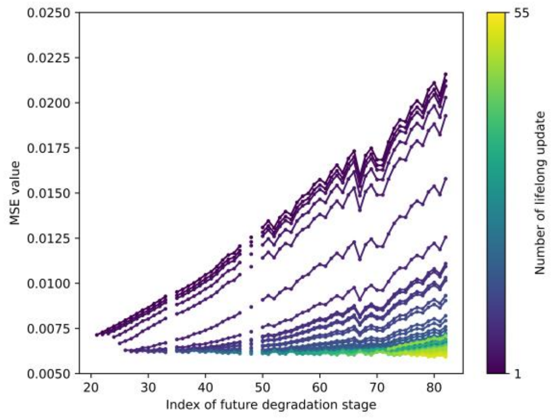
(a) Fine-tuned DT for charging



(b) AE-LSTM generated DT for charging



(c) Fine-tuned DT for discharging



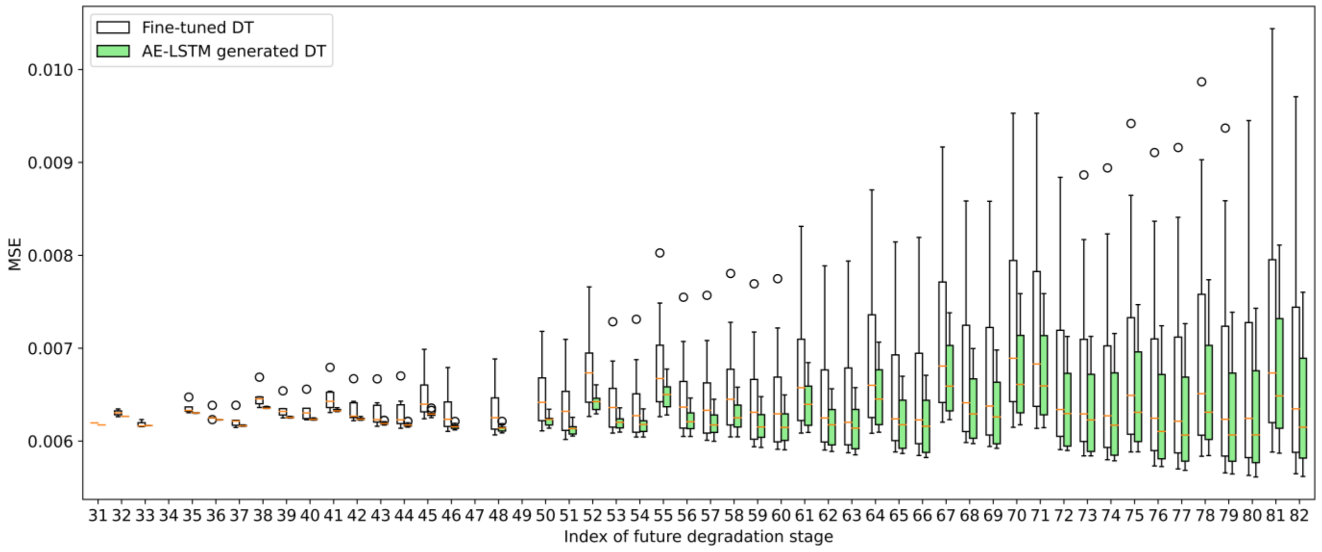
(d) AE-LSTM generated DT for discharging

Fig. 8. MSE values of the fine-tuned and AE-LSTM generated DT models with the battery dataset.

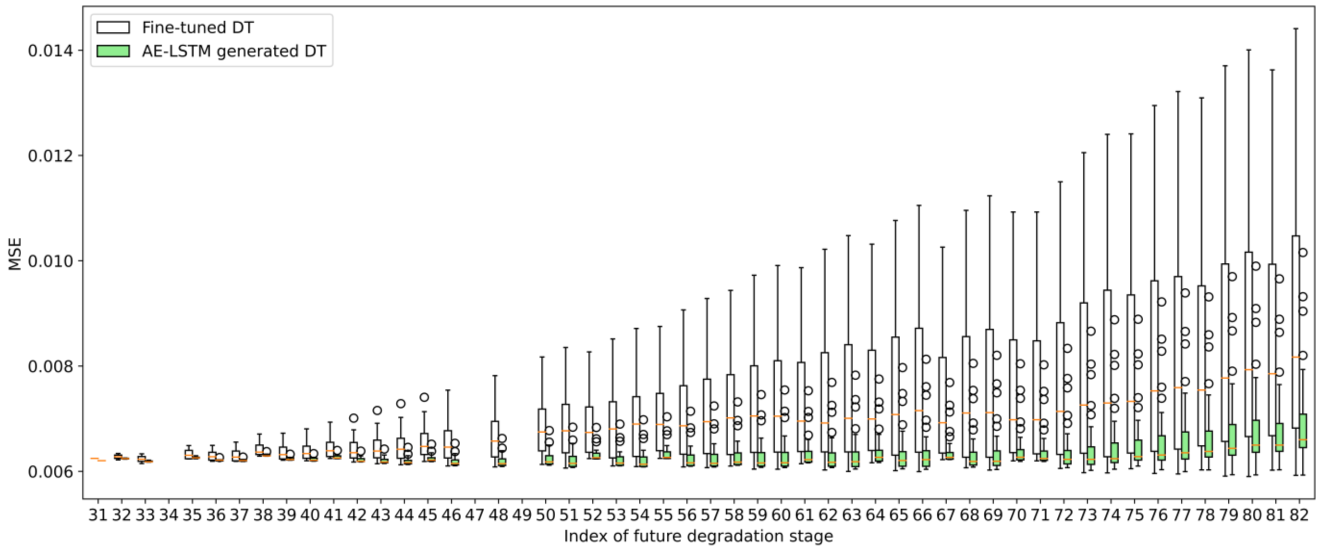
Apart from the above performance comparison, a success ratio is defined to compare the performance of fine-tuned and AE-LSTM generated DT models after completing a single lifelong update step. For instance, when the i-th lifelong update step is completed, the MSE values at all future degradation stages are obtained from the corresponding fine-tuned DT and the AE-LSTM generated DT. The success ratio sc_i at the i-th update step is defined as:

$$sc_i = \frac{n^i_{better}}{n^i_{all}} \tag{5}$$

where n^i_{better} is the number of future stages when the AE-LSTM generated DT has a smaller MSE value than the fine-tuned DT at the i-th update step. n^i_{all} is the number of all future degradation stages at the i-th update step.



(a) Charging

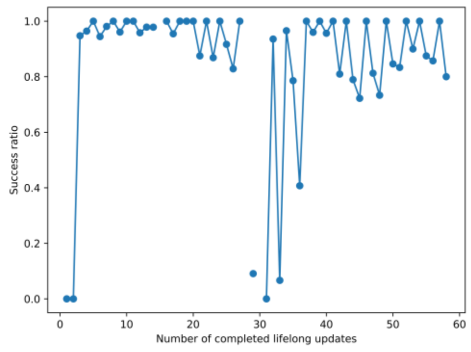


(b) Discharging

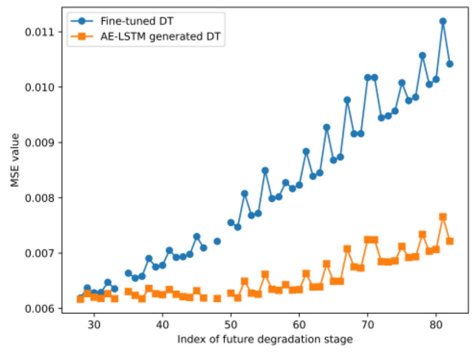
Fig. 9. Boxplot of MSE values at each stage during the lifelong update phase with the battery dataset.

The success ratios at all lifelong update steps when testing on the charging and discharging datasets are summarized in Fig. 10, along with MSE curves at the selected update steps. During the first several lifelong update steps, the success ratio values could be low and even close to zero. This means that the fine-tuned DT has a smaller MSE value than the proposed method at all future degradation stages. This is attributed to the insufficient warm-up phase, which fails to cover enough information on the dynamic behavior of the DT configuration.

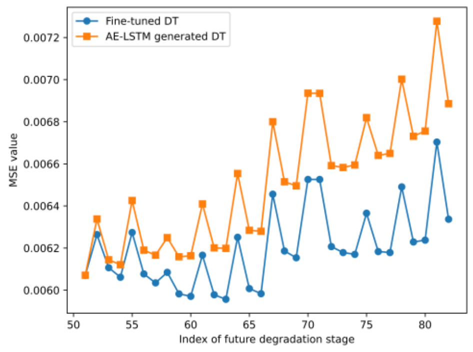
Although the low success ratio could sometimes be observed in the middle of the lifelong update phase, as shown in Fig. 10 (a) and (d), the difference between the MSE values obtained by fine-tuned and AE-LSTM generated DT models is quite small. For example, the maximum MSE difference between the two DT models after completing the 31st update is around 0.001 for battery charging in Fig. 10 (c), and the maximum difference after the 52nd update is around 0.0002 for battery discharging in Fig. 10 (f). It is observed that the success ratio values keep zeros at several successive lifelong update steps for battery discharging in Fig. 10 (d). The possible reason is that the AE-LSTM is trained with a constant hyperparameter setting, which would not provide better training performance at some update steps. However, at most lifelong update steps in both tests, the success ratio value is larger than 0.6, as shown in Fig. 10 (a) and (d). When the success ratio is close to one, the MSE value obtained by the proposed method is smaller than that obtained by the fine-tuned DT at each future degradation stage, as shown in Fig. 10 (b) and (e). Both observations demonstrate that the AE-LSTM generated DT could provide overall better predictions for future degradation stages than the fine-tuned DT method.



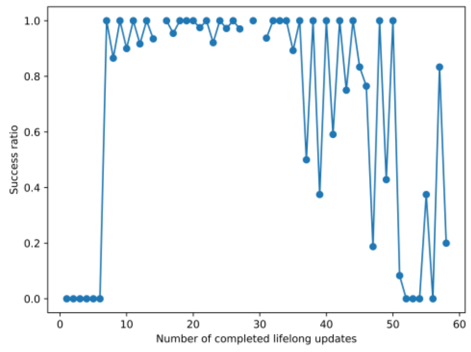
(a) Success ratio in charging



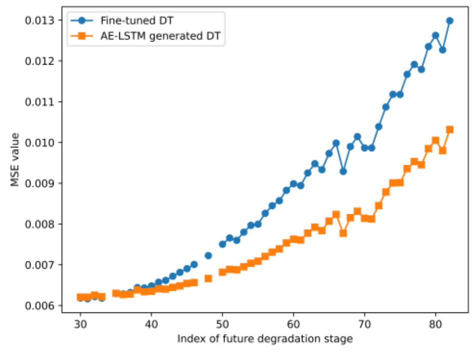
(b) MSE after the 8th update in charging



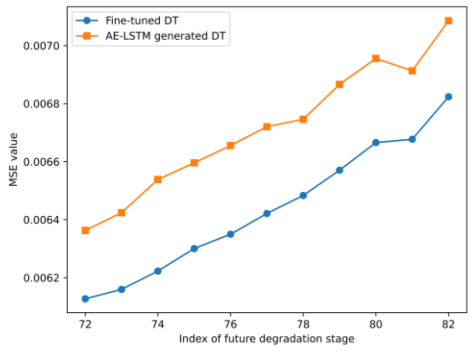
(c) MSE after the 31st update in charging



(d) Success ratio in discharging



(e) MSE after the 10th update in discharging



(f) MSE after the 52nd update in discharging

Fig. 10. Success ratio and comparison of MSE curves with the battery dataset.

Based on the above discussion, the proposed lifelong update method successfully enables the DT model to capture the effects of battery degradation on battery charging and discharging and outperforms the conventional fine-tuning method in terms of prediction performance at future unseen degradation stages in most lifelong update steps.

4 NASA ENGINE DATASET

4.1 Dataset description and preparation

The engine dataset covers a group of run-to-failure trajectories for a fleet of aircraft engines under real flight conditions, which are generated by the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) model developed at NASA [58]. During the data generation process, the flight conditions (i.e., altitude alt, flight mach number $Mach$, throttle-resolver angle TRA, and total temperature at fan inlet $T2$) recorded from a commercial jet are input to the CMAPSS model with the degradation of the engine components. Fourteen measurable time-series physical properties observed by sensors are simulated by the CMAPSS model, which outputs the estimated unobserved health parameters simultaneously. This process repeats as the number of degraded units increases until the health index of the engine reaches the end of its life.

There are eight datasets with different degradation units in the published repositories, where the health parameter degrades with the flight cycle number in each unit. This study is only tested on the data with the unit 1 in the file "DS04.h5", where 87 flight cycles are observed in the selected data. Fig. 11 summarizes the degradation of two health parameters over 87 flight cycles. According to Fig. 11, both the fan efficiency modifier and the fan flow modifier gradually degrade with the flight cycle. Meanwhile, the time-series total pressure p_1 at the fan outlet and the pressure p_2 at the low-pressure turbine outlet vary among different flight cycles, whose details are presented in [58].

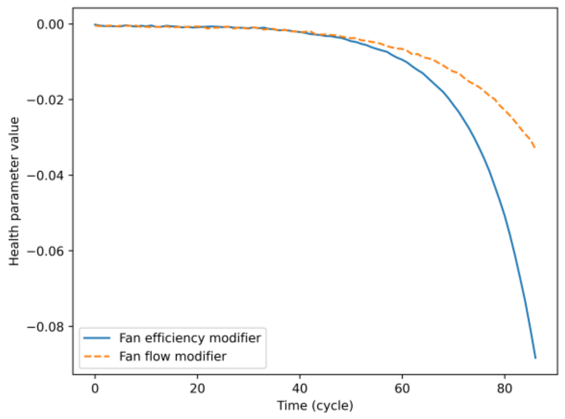


Fig. 11. Engine health indicator of unit 1 in dataset DS04.

In this test, the initial degradation stage is defined as the first flight cycle, and the future degradation stages are then defined chronologically, e.g., the stages T_0, T_1, T_2 are the flight cycles 1, 2, and 3, respectively. To evaluate the proposed lifelong update method, two DT models are constructed to predict the total pressures p_1 and p_2 respectively. The input variables of both DT models consist of four flight conditions (i.e., alt, $Mach$, TRA, $T2$) and the time t during one flight cycle. The output is the corresponding pressure p at the time t under the given flight condition. In this dataset, the number of data points on the raw pressure curve varies from 4700 to 12500 during 87 flight cycles. To reduce the nonlinearity in the raw dataset, both input and output data are smoothed by the moving average filter with a window size of 50. The down-sampling with the same window size of 50 is then performed to reduce the number of data points. Finally, all data of each input or output variable at all flight cycles are scaled simultaneously by the Minmax scaler to the range [0,1]. The scaled datasets are used for DT modeling, fine-tuning, and training dynamic models in the proposed method.

4.2 Test setting

Considering that the nonlinearity in the engine datasets is higher than that in the battery dataset, a deeper FNN structure is used for DT models. Apart from the input and output layers, the applied structure has seven hidden layers with the activation function Tanh(·), whose dimensions are detailed in Fig. 12. The training process of models DT_0 at the initial degradation stage T_0 for both selected pressures is the same as that in Section 3.2.

As the applied FNN structure is deeper in this test, the entropy values of $\boldsymbol{\theta}_0$ in the two initial DT models for both pressures are around 12.9, and the dimensions of the latent space in autoencoders are determined as 26 based on the predefined hyperparameters. During the fine-tuning process of the DT model at each degradation stage, and the training process of the autoencoder and the LSTM, the hyperparameter settings are identical to those in Table 2. The only difference is that the number m of stages in the warm-up phase and the window size w for input sequences of the LSTM are updated as $m = 40$ and $w = 10$ for the engine dataset.

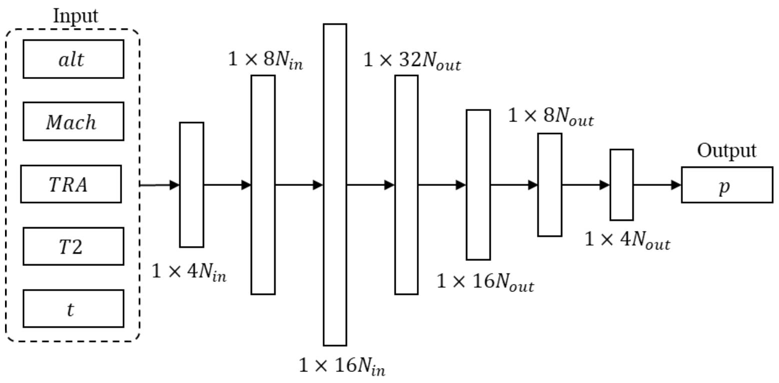
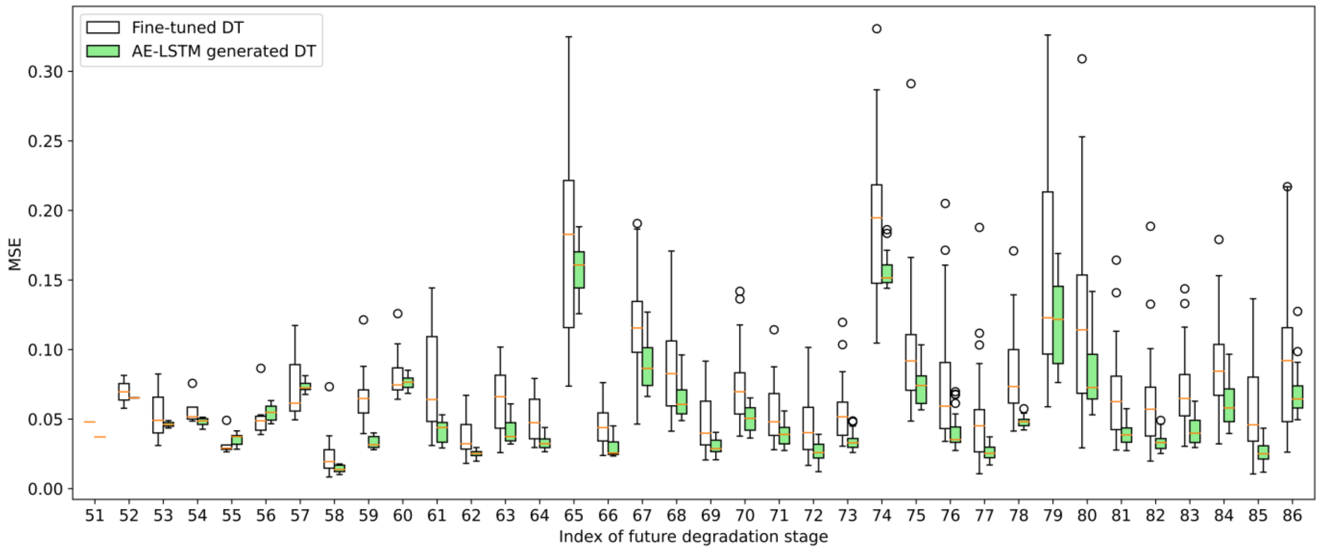


Fig. 12. FNN structure for the NASA engine dataset. The input contains the altitude alt, flight Mach number $Mach$, throttle-resolver angle TRA, total temperature at fan inlet $T2$, and time t. The output is the pressure p. The input and output dimensions are $N_{in} = 5$ and $N_{out} = 1$, respectively.

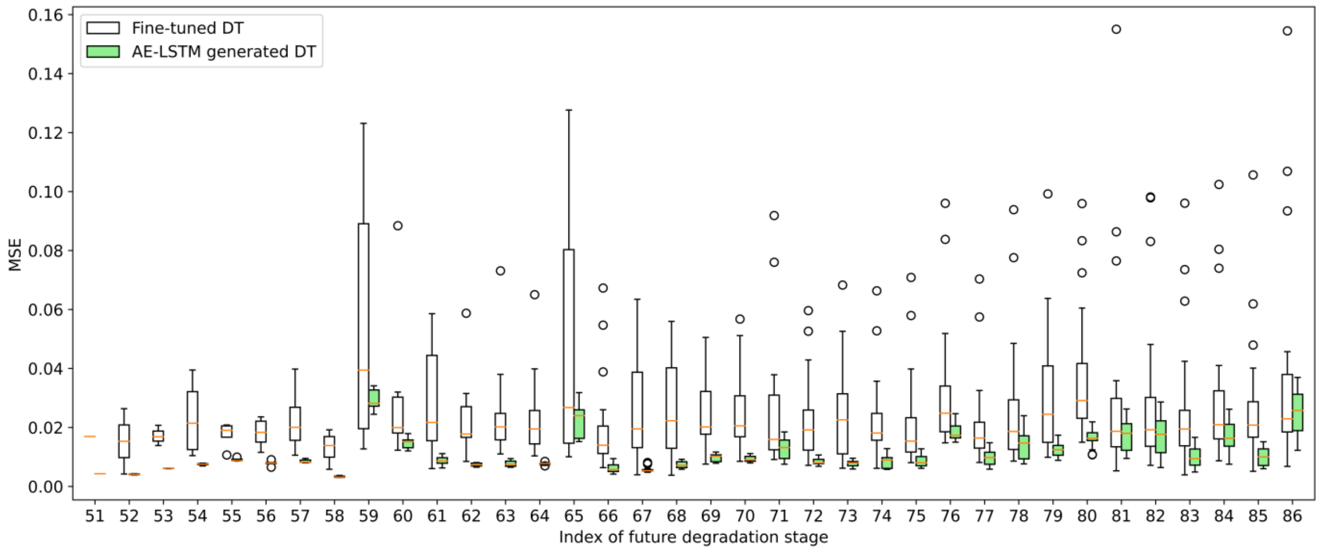
4.3 Results

When assessing the proposed method in the engine dataset, the obtained results are analyzed with the same setting as the battery dataset in Section 3. The boxplots of MSE values obtained by the two compared DT models at each degradation stage during the lifelong update process are shown in Fig. 13, with the same setting as for the battery dataset test. The comparison results show that the AE-LSTM generated DT has smaller median MSE values and smaller box sizes at each degradation stage. This observation indicates that the proposed lifelong update method outperforms the conventional fine-tuning method in terms of both prediction accuracy and robustness for future unseen stages.

The success ratio calculated at each lifelong update during the test is shown in Fig. 14. At most lifelong updates, the value is larger than 0.5 for p_1 and p_2, as shown in Fig. 14 (a) and (d). Meanwhile, the success ratio increases with the number of completed lifelong update steps. Although the small ratio value is observed at the first several updates for both pressures in Fig. 14 (b) and (e), the value could increase to 1 at future update, when the AE-LSTM generated DT outperforms the fine-tuned DT at each future degradation stage as shown in Fig. 14 (c) and (f). Therefore, with the data from more degradation stages, the proposed lifelong update method would learn more information about the dynamic behavior of model configurations during the lifecycle.



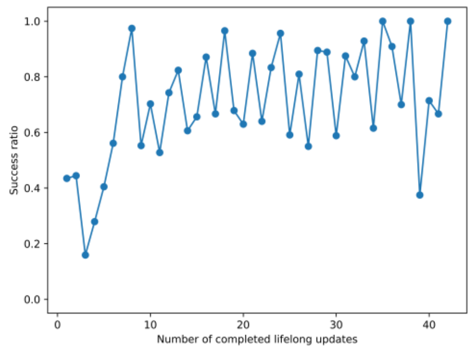
(a) Total pressure p_1 at the fan outlet



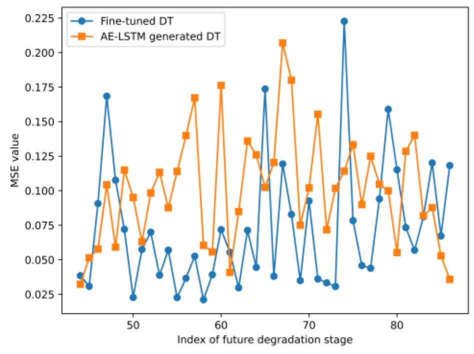
(b) Total pressure p_2 at the LPT outlet

Fig. 13. Boxplot of MSE values at each stage during the lifelong update phase for the engine dataset.

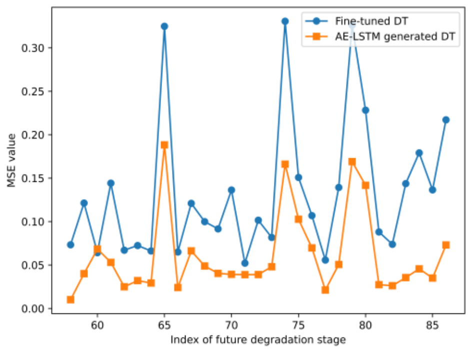
Based on those discussions, the proposed lifelong update method could identify the effects of engine degradation on engine responses by learning the trend of model configurations during the lifecycle. Moreover, compared with the conventional fine-tuning framework, the proposed lifelong update method could provide better prediction performance for future unseen degradation stages in most cases.



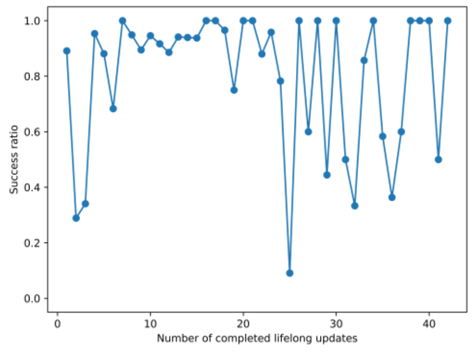
(a) Success ratio in p_1



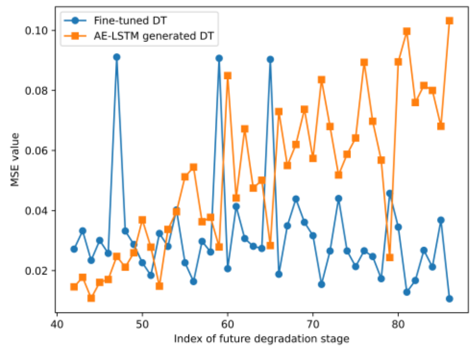
(b) MSE after the 4th update in p_1



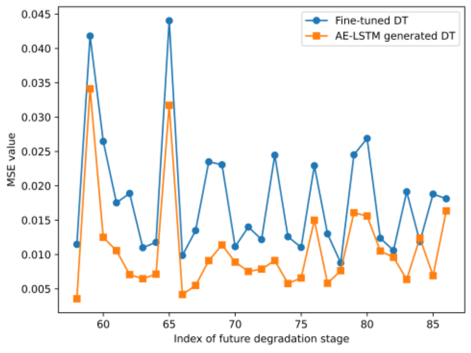
(c) MSE after the 18th update in p_1



(d) Success ratio in p_2



(e) MSE after the 2nd update in p_2



(f) MSE after the 18th update in p_2

Fig. 14. Success ratio and comparison of MSE curves for the engine dataset.

5 DISCUSSIONS

This study focuses on updating a given DT model (i.e., neural network in this study), rather than developing new modeling methods. By representing system degradation by the temporal evolution of DT model parameters, the proposed framework effectively captures degradation effects and enables accurate prediction at unseen lifecycle stages. Since no prior knowledge of degradation mechanisms is assumed, conventional degradation modeling approaches are not applicable in this work. Therefore, the standard fine-tuning is used as the primary baseline for comparison. In addition, related modeling components integrated into the framework are evaluated to examine the generalization and applicability of the proposed concept.

5.1 Efficiency, robustness, generalization, and sensitivity

This section summarizes performance studies of the proposed framework in terms of efficiency, robustness, sensitivity, and generalization. The corresponding selected results are presented in the appendices. More detailed results of various tests are accessible at https://github.com/tyf0416/Result-summary-for-DT-lifelong-update.

(a) Efficiency. The proposed lifelong update framework is computationally efficient and suitable for online deployment. Across both the battery and engine degradation datasets, fine-tuning a model requires only a fraction of a second per lifecycle, and the time to train AE-LSTM is less than one minute on a commodity desktop. Both times are negligible relative to the physical system cycle duration. Therefore, the proposed approach enables lifelong DT adaptation without imposing a computational burden during the system lifecycle. The detailed timing analysis is presented in Appendix 0.

(b) Robustness. In Appendix C, a two-stage variance analysis shows that the proposed lifelong update framework is highly stable with respect to stochasticity in the update process, while most performance variability originates from the initialization of the digital twin model. When the same initial DT is used, the lifelong updates produce consistently low variance across degradation stages for both battery and engine datasets, with only minor fluctuations despite stochastic training. In contrast, different initial DT model seeds lead to distinct convergence levels, indicating that initialization is the dominant source of uncertainty. These results confirm that the proposed update mechanism itself is robust and reproducible once a reasonable initial DT is established.

(c) Generalization. The proposed framework demonstrates strong generalization across different time-series forecasting models and feature extraction strategies. Across both battery and engine datasets, AE-LSTM, AE-GRU, and AE-Transformer consistently outperform conventional fine-tuning, with LSTM providing the best overall trade-off between accuracy and robustness. Nonlinear autoencoder-based feature extraction significantly outperforms linear and layer-wise alternatives, confirming the importance of learning joint, nonlinear latent representations of DT parameters. Meanwhile, the presented autoencoder structure achieves the best

generalization, while overly simple or deep structures lead to under- and over-fitting, respectively. Moreover, the variational autoencoders can further improve performance when a small KL-regularization weight is used, demonstrating that probabilistic latent representations can be effectively integrated into the proposed framework. More detailed comparisons are presented in Appendixes D-E.

(d) Sensitivity. The framework exhibits predictable and stable behavior with respect to key hyperparameters governing latent representation and temporal modeling. Generally, larger latent dimensions improve prediction accuracy and robustness, particularly when they exceed the intrinsic entropy of the initial DT model. Moderate LSTM window sizes achieve the best balance between temporal context and predictive stability, whereas overly large windows degrade performance. Meanwhile, longer warm-up phases consistently improve accuracy and robustness by providing richer historical degradation information. These trends provide practical guidance for configuring the proposed framework in real-world applications. A detailed analysis is discussed in Appendix F.

5.2 Limitations and future work

Although some hyperparameters have been discussed, a comprehensive hyperparameter optimization is not performed for all components, such as the DT and LSTM structures. Based on findings on different autoencoder structures, similar trends could be anticipated in LSTM. A simpler LSTM may suffer from underfitting and fail to capture the complex temporal evolution of the latent features of DT model configurations. In contrast, a deeper LSTM model with more layers may suffer from overfitting, resulting in poor generalization to unseen future degradation stages. Therefore, hyperparameter optimization would be valuable for identifying a better initial model or improving the performance of the AE-LSTM.

Meanwhile, this work only compares with the conventional fine-tuning. Although the optimization based on performance differences is the most commonly applied model updating method to update DT parameters [59,60], there are some other potential update methods to consider the lifecycle information. For instance, the discrepancy modeling method trains a model to learn the error between DT predictions and real measurements, and the trained model is combined with the original DT model as the final updated one [61]. A broader comparison with different model updating techniques in the artificial intelligence field will be a natural extension of this work.

The benefits of the proposed method have been demonstrated by real-world battery and engine datasets, while the degradation behavior exists in many practical systems, such as bearing [62], gear [19], and infrastructure [20]. The generalizability and applicability of the proposed lifelong update method need more validation on lifecycle datasets from various engineering fields. For instance, the quality of parts printed by additive manufacturing systems may decrease after a long operation period. The proposed framework could update the process model to capture the quality degradation behavior, which is used to provide better feedback in the model predictive control. In an infrastructure (e.g., bridges), the proposed framework could be adopted to periodically update the neural network-based structural property model from the sensor measurements, which serves predictive maintenance.

When deploying the proposed lifelong update method to practical systems, several potential challenges exist.

(a) *Dynamic nonlinearity and non-stationarity in data*. The constant DT model structure may fail to capture the dynamic nonlinearity and non-stationarity of data among degradation stages in the lifecycle. In such cases, the optimal model structure evolves with the degradation stage. A similar case is observed in the electro-optical system, where a nonparametric Bayesian graph neural network is constructed to denote the dynamic degradation process of health stage and the propagation of epistemic uncertainty [14]. When the data at new time steps are observed, the Dirichlet process mixture model and the Gaussian particle filter are adopted to update the optimal network structure and obtain the new model parameters, respectively. However, the work [14] only updates the model step by step, still failing to find the optimal model structure at future time steps in advance. Therefore, one future work is to find a method to represent the dynamic behavior of optimal model structures and corresponding parameters simultaneously over the degradation stages in the lifecycle, which enables predicting the optimal configuration (i.e., structure and parameters) at unseen future stages.

(b) *Dynamic data quality and size*. Among degradation stages, the data quality and size may change according to operation and monitoring conditions. Optimization is then required to find the optimal training hyperparameters for each degradation stage, increasing deployment cost. In the work, hyperparameters are constant through the lifecycle, which contribute to the observation that AE-LSTM achieves worse performance at some degradation stages during the latter half lifelong update phase. Therefore, how to choose the optimal hyperparameters automatically for learning at each stage would be another future work to make the DT provide stable and better performance during the lifelong update.

(c) *Sudden degradation in actual systems*. According to reference [41], system degradations could follow different patterns, such as incremental or sudden changes. The proposed method is designed for incremental degradation, where the performance distribution slowly changes in a continuous manner. However, if a sudden failure occurs due to unexpected environmental conditions or inappropriate operations, the system's behavior changes abruptly. In such a case, the AE-LSTM may fail to track this sharp transition, resulting in poor prediction performance for future degradation stages. This problem may be solved by concept drift detection methods [63] to identify the changing point of degradation, and reinitializing the time-series forecasting methods for each incremental degradation stage.

6 SUMMARY

For systems degradation during a lifecycle, this paper proposes a lifelong update method for corresponding digital twin (DT) models. Different from DT models relying on degradation parameters or online update methods focusing on performance at one degradation stage, the proposed method aims to ensure that the DT model remains valid through all unseen future degradation stages. To fulfill this purpose, the system degradation is represented as the temporal behavior of the digital twin configurations at continuous degradation stages. In this work, a feedforward neural network with a fixed structure is adopted as the DT model, which is fine-tuned at each degradation stage. The optimized network parameters at known stages are used to train an autoencoder model to learn the latent features of high-dimensional model parameters. A long short-term memory (LSTM) model is then adopted to identify the dynamic behavior of latent features over degradation stages. Once both models are trained, the latent features at any future degradation stage are estimated by LSTM, based on which the model parameters are reconstructed by the decoder to predict the system responses. Benefits of the proposed method have been demonstrated by test results in two real-world engineering datasets, i.e., battery degradation data and flight engine dataset. Compared with the conventional fine-tuning method, the proposed method can provide predictions with better accuracy and robustness at future unseen degradation stages.

A comprehensive discussion is presented on the efficiency, robustness, generalization, and sensitivity of the proposed method, yielding the following conclusions.

- The lifelong update method provides stable updated DT models when the initial DT model is given.
- The proposed method is compatible with different forecasting architectures, as demonstrated by performance among Gated Recurrent Unit, Transformer, and LSTM models.
- Processing parameters of all hidden layers of the DT model simultaneously with nonlinear feature extraction methods facilitates finding latent features involving relationships among hidden layers, which in turn provides better final prediction performance at future stages.
- The max entropy of the initial DT model is a principle lower bound of the latent feature dimension in autoencoder. When the latent feature dimension is smaller than the max entropy, the autoencoder lacks sufficient capability to capture and reconstruct the underlying degradation pattern of DT model parameters through the lifecycle.

With the proposed method, a DT model is able to identify how the system responses are affected by system degradation during the lifecycle, even though the system is far from its end of life. The proposed method provides an economical alternative to expensive aging experiments and enables a system to deploy directly without prior aging tests. This would bring economic benefits to system manufacturers. Meanwhile, it can predict the responses of degradation-affected systems at unknown future stages, which is beneficial to conducting health management and monitoring.

Although key hyperparameters are analyzed in the work, broader optimization such as model architecture search remains unexplored currently. Meanwhile, generalizability to other complex systems requires further validation across diverse domains, e.g., additive manufacturing, and infrastructure health monitoring. Several challenges remain open when deploying to practical systems for future research. (a) The nonlinearity and non-stationarity in the data changes over the lifecycle, requiring to find the optimal model structures and parameters per stage, whose dynamic patterns should be captured jointly to consider their interdependencies. (b) The data quality and size change through the lifecycle, requiring automatically adapting training hyperparameters per stage for better learning performance. (c) Sudden degradation events would hinder the time-series model from providing accurate predictions for future stages, which could be tackled by concept drift detection methods and adaptive re-initialization of time-series models. Addressing these aspects represents a promising future direction toward fully self-updating, self-configuring DT models deployable in real industrial systems.

CRediT author contribution statement

Yifan Tang: Conceptualization, Methodology, Data curation, Writing - Original Draft, Writing - Review & Editing, Visualization; **Mostafa Rahmani Dehaghani**: Conceptualization, Methodology; **G. Gary Wang**: Conceptualization, Methodology, Supervision, Writing - Review & Editing, Project administration

Declaration of interest statement

The authors declare that they have no conflicts of interest.

Data availability

The open-source datasets in the work can be found from the cited references.

Acknowledgements

Funding from Natural Sciences and Engineering Research Council (NSERC) of Canada under the project RGPIN-2019-06601, as well Graduate Dean's Entrance Scholarship (GDES) from Simon Fraser University, is

appreciated.

References

[1] V. Karkaria, Y.K. Tsai, Y.P. Chen, W. Chen, An optimization-centric review on integrating artificial intelligence and digital twin technologies in manufacturing, Engineering Optimization 57 (2025) 161–207. https://doi.org/10.1080/0305215X.2024.2434201.

[2] A. Thelen, X. Zhang, O. Fink, Y. Lu, S. Ghosh, B.D. Youn, M.D. Todd, S. Mahadevan, C. Hu, Z. Hu, A comprehensive review of digital twin — part 1: modeling and twinning enabling technologies, Structural and Multidisciplinary Optimization 65 (2022). https://doi.org/10.1007/s00158-022-03425-4.

[3] F. Xiang, Z. Zhang, Y. Zuo, F. Tao, Digital twin driven green material optimal-selection towards sustainable manufacturing, Procedia CIRP 81 (2019) 1290–1294. https://doi.org/10.1016/j.procir.2019.04.015.

[4] X. Zhu, Y. Ji, A digital twin–driven method for online quality control in process industry, International Journal of Advanced Manufacturing Technology 119 (2022) 3045–3064. https://doi.org/10.1007/s00170-021-08369-5.

[5] L. Li, S. Aslam, A. Wileman, S. Perinpanayagam, Digital twin in aerospace industry: a gentle introduction, IEEE Access 10 (2022) 9543–9562. https://doi.org/10.1109/ACCESS.2021.3136458.

[6] S. Su, R.Y. Zhong, Y. Jiang, J. Song, Y. Fu, H. Cao, Digital twin and its potential applications in construction industry: state-of-art review and a conceptual framework, Advanced Engineering Informatics 57 (2023) 102030. https://doi.org/10.1016/j.aei.2023.102030.

[7] B. Zhang, G. Ding, Q. Zheng, K. Zhang, S. Qin, Iterative updating of digital twin for equipment: progress, challenges, and trends, Advanced Engineering Informatics 62 (2024) 102773. https://doi.org/10.1016/j.aei.2024.102773.

[8] W. Qian, M. Tang, H. Gao, J. Dong, J. Liang, J. Liu, Improving indoor air flow and temperature prediction with local measurements based on CFD-EnKF data assimilation, Building and Environment 223 (2022) 109511. https://doi.org/10.1016/j.buildenv.2022.109511.

[9] S. Eftekhar Azam, S. Mariani, Online damage detection in structural systems via dynamic inverse analysis: a recursive Bayesian approach, Engineering Structures 159 (2018) 28–45. https://doi.org/10.1016/j.engstruct.2017.12.031.

[10] S. Ereiz, I. Duvnjak, J. Fernando Jiménez-Alonso, Review of finite element model updating methods for structural applications, Structures 41 (2022) 684–723. https://doi.org/10.1016/j.istruc.2022.05.041.

[11] S. Cheng, C. Quilodran-Casas, S. Ouala, A. Farchi, C. Liu, P. Tandeo, R. Fablet, D. Lucor, B. Iooss, J. Brajard, D. Xiao, T. Janjic, W. Ding, Y. Guo, A. Carrassi, M. Bocquet, R. Arcucci, Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review, IEEE/CAA Journal of Automatica Sinica 10 (2023) 1361–1387. https://doi.org/10.1109/JAS.2023.123537.

[12] S. Al-Subaihawi, J.M. Ricles, S.E. Quiel, Online explicit model updating of nonlinear viscous dampers for real time hybrid simulation, Soil Dynamics and Earthquake Engineering 154 (2022) 107108. https://doi.org/10.1016/j.soildyn.2021.107108.

[13] G.B. Huang, N.Y. Liang, H.J. Rong, P. Saratchandran, N. Sundararajan, On-line sequential extreme learning machine, in: IASTED International Conference on Computational Intelligence, ACTA Press, Calgary, Canada, 2005: pp. 232–237.

[14] J. Yu, Y. Song, D. Tang, J. Dai, A digital twin approach based on nonparametric Bayesian network for complex system health monitoring, Journal of Manufacturing Systems 58 (2021) 293–304. https://doi.org/10.1016/j.jmsy.2020.07.005.

[15] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, Y. Wang, Dynamic neural networks: a survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2022) 7436–7456. https://doi.org/10.1109/TPAMI.2021.3117837.

[16] W. Li, Y. Li, A. Garg, L. Gao, Enhancing real-time degradation prediction of lithium-ion battery: A digital twin framework with CNN-LSTM-attention model, Energy 286 (2024) 129681. https://doi.org/10.1016/j.energy.2023.129681.

[17] J. Shi, A. Rivera, D. Wu, Battery health management using physics-informed machine learning: Online degradation modeling and remaining useful life prediction, Mechanical Systems and Signal Processing 179 (2022) 109347. https://doi.org/10.1016/j.ymssp.2022.109347.

[18] X. Yang, Y. Ran, G. Zhang, H. Wang, Z. Mu, S. Zhi, A digital twin-driven hybrid approach for the prediction of performance degradation in transmission unit of CNC machine tool, Robotics and Computer-Integrated Manufacturing 73 (2022) 102230. https://doi.org/10.1016/j.rcim.2021.102230.

[19] B. He, L. Liu, D. Zhang, Digital twin-driven remaining useful life prediction for gear performance degradation: A review, Journal of Computing and Information Science in Engineering 21 (2021) 030801. https://doi.org/10.1115/1.4049537.

[20] G. Prakash, X.X. Yuan, B. Hazra, D. Mizutani, Toward a big data-based approach: a review on degradation models for prognosis of critical infrastructure, Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems 4 (2021) 021005. https://doi.org/10.1115/1.4048787.

[21] P. Paris, F. Erdogan, A critical analysis of crack propagation laws, Journal of Fluids Engineering, Transactions of the ASME 85 (1963) 528–533. https://doi.org/10.1115/1.3656900.

[22] M. Jia, Z. Wu, X. Jiang, R.C. Yu, X. Zhang, Y. Wang, Modified Paris law for mode I fatigue fracture of concrete based on crack propagation resistance, Theoretical and Applied Fracture Mechanics 131 (2024) 104383. https://doi.org/10.1016/j.tafmec.2024.104383.

[23] M.U. Khan, S. Ahmad, H.J. Al-Gahtani, Chloride-induced corrosion of steel in concrete: an overview on chloride diffusion and prediction of corrosion initiation time, International Journal of Corrosion 1 (2017) 5819202. https://doi.org/10.1155/2017/5819202.

[24] X. Li, X. Lin, H. Zhu, X. Wang, Z. Liu, Condition assessment of shield tunnel using a new indicator: the tunnel serviceability index, Tunnelling and Underground Space Technology 67 (2017) 98–106. https://doi.org/10.1016/j.tust.2017.05.007.

[25] L. Bellini Machado, M. Massao Futai, Tunnel performance prediction through degradation inspection and digital twin construction, Tunnelling and Underground Space Technology 144 (2024) 105544. https://doi.org/10.1016/j.tust.2023.105544.

[26] M. Yue, K. Benaggoune, J. Meng, D. Diallo, Implementation of an early stage fuel cell degradation prediction digital twin based on transfer learning, IEEE Transactions on Transportation Electrification 9 (2023) 3308–3318. https://doi.org/10.1109/TTE.2022.3229716.

[27] J. Yu, Adaptive hidden Markov model-based online learning framework for bearing faulty detection and performance degradation monitoring, Mechanical Systems and Signal Processing 83 (2017) 149–162. https://doi.org/10.1016/j.ymssp.2016.06.004.

[28] H. Li, C. Chen, J. Wei, Z. Chen, G. Lei, L. Wu, State of health (SOH) estimation of lithium-ion batteries based on ABC-BiGRU, Electronics 13 (2024) 1675. https://doi.org/10.3390/electronics13091675.

[29] J. Sun, Z. Yan, Y. Han, X. Zhu, C. Yang, Deep learning framework for gas turbine performance digital twin and degradation prognostics from airline operator perspective, Reliability Engineering and System Safety 238 (2023) 109404. https://doi.org/10.1016/j.ress.2023.109404.

[30] J. Dai, L. Tian, T. Han, H. Chang, Digital twin for wear degradation of sliding bearing based on PFENN, Advanced Engineering Informatics 61 (2024) 102512. https://doi.org/10.1016/j.aei.2024.102512.

[31] C. Chen, H. Fu, Y. Zheng, F. Tao, Y. Liu, The advance of digital twin for predictive maintenance: the role and function of machine learning, Journal of Manufacturing Systems 71 (2023) 581–594. https://doi.org/10.1016/j.jmsy.2023.10.010.

[32] X. Qu, Y. Song, D. Liu, X. Cui, Y. Peng, Lithium-ion battery performance degradation evaluation in dynamic operating conditions based on a digital twin model, Microelectronics Reliability 114 (2020) 113857. https://doi.org/10.1016/j.microrel.2020.113857.

[33] X. Tang, X. Qin, K. Wei, S. Xu, A novel online degradation model for proton exchange membrane fuel cell based on online transfer learning, International Journal of Hydrogen Energy 48 (2023) 13617–13632. https://doi.org/10.1016/j.ijhydene.2022.12.260.

[34] S. Chen, S. Wang, P. Wen, S. Zhao, Digital twin for degradation parameters identification of DC-DC converters based on Bayesian optimization, in: 2021 IEEE International Conference on Prognostics and Health Management, ICPHM 2021, IEEE, Detroit (Romulus), MI, USA, 2021: pp. 1–9. https://doi.org/10.1109/ICPHM51084.2021.9486446.

[35] Q. Wu, W. Wang, Q. Wang, L. Xiao, B. Hu, Digital twin approach for degradation parameters identification of a single-phase DC-AC inverter, in: IEEE Applied Power Electronics Conference and Exposition, IEEE, Houston, TX, USA, 2022: pp. 1725–1730. https://doi.org/10.1109/APEC43599.2022.9773462.

[36] L. Xiong, Y. He, Y. Chen, J. Lu, G. Niu, Digital twin-based degradation prediction for train electro-pneumatic valve, Reliability Engineering and System Safety 240 (2023) 109627. https://doi.org/10.1016/j.ress.2023.109627.

[37] K. Feng, J.C. Ji, Y. Zhang, Q. Ni, Z. Liu, M. Beer, Digital twin-driven intelligent assessment of gear surface degradation, Mechanical Systems and Signal Processing 186 (2023) 1–23. https://doi.org/10.1016/j.ymssp.2022.109896.

[38] K. Feng, W.A. Smith, R.B. Randall, H. Wu, Z. Peng, Vibration-based monitoring and prediction of surface profile change and pitting density in a spur gear wear process, Mechanical Systems and Signal Processing 165 (2022) 108319. https://doi.org/10.1016/j.ymssp.2021.108319.

[39] Y. Yi, C. Xia, C. Feng, W. Zhang, C. Fu, L. Qian, S. Chen, Digital twin-long short-term memory (LSTM) neural network based real-time temperature prediction and degradation model analysis for lithium-ion battery, Journal of Energy Storage 64 (2023) 107203. https://doi.org/10.1016/j.est.2023.107203.

[40] A. Inamdar, M. van Soestbergen, A. Mavinkurve, W.D. van Driel, G.Q. Zhang, Modelling thermomechanical degradation of moulded electronic packages using physics-based digital twin, Microelectronics Reliability 157 (2024) 115416. https://doi.org/10.1016/J.MICROREL.2024.115416.

[41] F. Bayram, B.S. Ahmed, A. Kassler, From concept drift to model degradation: An overview on performance-aware drift detectors, Knowledge-Based Systems 245 (2022) 108632. https://doi.org/10.1016/j.knosys.2022.108632.

[42] K. Hasegawa, K. Fukami, T. Murata, K. Fukagata, Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes, Theoretical and Computational Fluid Dynamics 34 (2020) 367–383. https://doi.org/10.1007/s00162-020-00528-w.

[43] Y. Kiarashinejad, S. Abdollahramezani, A. Adibi, Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures, Npj Computational Materials 6 (2020) 1–12. https://doi.org/10.1038/s41524-020-0276-y.
[44] P. Li, Y. Pei, J. Li, A comprehensive survey on design and application of autoencoder in deep learning, Applied Soft Computing 138 (2023) 110176. https://doi.org/10.1016/j.asoc.2023.110176.
[45] L.V.N. Lapenda, R.P. Monteiro, C.J.A. Bastos-Filho, Autoencoder latent space: an empirical study, in: 2020 IEEE Symposium Series on Computational Intelligence, IEEE, Canberra, ACT, Australia, 2020: pp. 2453–2460. https://doi.org/10.1109/SSCI47803.2020.9308551.
[46] K. Mai Ngoc, M. Hwang, Finding the best k for the dimension of the latent space in autoencoders, in: Computational Collective Intelligence: 12th International Conference, ICCCI 2020, Springer International Publishing, Da Nang, Vietnam, November 30–December 3, 2020, 2020: pp. 453–464. https://doi.org/10.1007/978-3-030-63007-2_35.
[47] M. Yuan, H. Hoskens, S. Goovaerts, N. Herrick, M.D. Shriver, S. Walsh, P. Claes, Hybrid autoencoder with orthogonal latent space for robust population structure inference, Scientific Reports 13 (2023) 1–14. https://doi.org/10.1038/s41598-023-28759-x.
[48] J.A. Dabin, A.M. Haimovich, J. Mauger, A. Dong, Blind source separation with L1 regularized sparse autoencoder, in: 2020 29th Wireless and Optical Communications Conference, WOCC 2020, IEEE, Newark, NJ, USA, 2020: pp. 5–9. https://doi.org/10.1109/WOCC48579.2020.9114943.
[49] D.A. Lelewer, D.S. Hirschberg, Data compression, ACM Computing Surveys (CSUR) 19 (1987) 261–296. https://doi.org/10.1145/45072.45074.
[50] A. Somazzi, P. Ferragina, D. Garlaschelli, On nonlinear compression costs: when Shannon meets Rényi, IEEE Access 12 (2024) 77750–77763. https://doi.org/10.1109/ACCESS.2024.3406912.
[51] H.S. Seung, H. Sompolinsky, Statistical mechanics of learning from examples, Physical Review A 45 (1992) 6056.
[52] G. Van Houdt, C. Mosquera, G. Nápoles, A review on the long short-term memory model, Artificial Intelligence Review 53 (2020) 5929–5955. https://doi.org/10.1007/s10462-020-09838-1.
[53] A.H. Bukhari, M.A.Z. Raja, M. Sulaiman, S. Islam, M. Shoaib, P. Kumam, Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting, IEEE Access 8 (2020) 71326–71338. https://doi.org/10.1109/ACCESS.2020.2985763.
[54] M. Awais, M. Raza, N. Singh, K. Bashir, U. Manzoor, S.U. Islam, J.J.P.C. Rodrigues, LSTM-based emotion detection using physiological signals: IoT framework for healthcare and distance learning in COVID-19, IEEE Internet of Things Journal 8 (2021) 16863–16871. https://doi.org/10.1109/JIOT.2020.3044031.
[55] B. Lindemann, B. Maschler, N. Sahlab, M. Weyrich, A survey on anomaly detection for technical systems using LSTM networks, Computers in Industry 131 (2021) 103498. https://doi.org/10.1016/j.compind.2021.103498.
[56] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.L. Boulesteix, D. Deng, M. Lindauer, Hyperparameter optimization: foundations, algorithms, best practices, and open challenges, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 13 (2023) 1–43. https://doi.org/10.1002/widm.1484.
[57] C. Birkl, Diagnosis and prognosis of degradation in lithium-ion batteries, University of Oxford, 2017.
[58] M.A. Chao, C. Kulkarni, K. Goebel, O. Fink, Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics, Data 6 (2021) 1–14. https://doi.org/10.3390/data6010005.
[59] W. Kim, B.D. Youn, Physics-based digital twin updating and twin-based explainable crack identification of mechanical lap joint, Reliability Engineering and System Safety 253 (2025) 110515. https://doi.org/10.1016/j.ress.2024.110515.
[60] R.R. Rabi, G. Monti, Genetic Algorithm-Based Model Updating in a Real-Time Digital Twin for Steel Bridge Monitoring, Applied Sciences 15 (2025) 4074. https://doi.org/10.3390/app15084074.
[61] J. Chen, C. Luey, C. Siuta, D. Lim, R. Radulescu, W. Chen, A Digital Twin Framework Utilizing Machine Learning for Robust Predictive Maintenance : Enhancing Tire Health Monitoring, Journal of Computing and Information Science in Engineering 25 (2025) 071003. https://doi.org/10.1115/1.4067270.
[62] F. Cartella, H. Sahli, Online adaptive bearings condition assessment using continuous hidden Markov models, Advances in Mechanical Engineering 7 (2015). https://doi.org/10.1155/2014/758785.
[63] M. Myuu, W. Yan, Accurate detecting concept drift in evolving data streams, ICT Express 6 (2020) 332–338. https://doi.org/10.1016/j.icte.2020.05.011.
[64] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, L. Sun, Transformers in time series: a survey, in: IJCAI International Joint Conference on Artificial Intelligence, Macao, S.A.R, 2023: pp. 6778–6786. https://doi.org/10.24963/ijcai.2023/759.
[65] M. Waqas, U.W. Humphries, A Critical Review of RNN and LSTM Variants in Hydrological Time Series Predictions, MethodsX 13 (2024) 102946. https://doi.org/10.1016/j.mex.2024.102946.
[66] M. Ławryńczuk, K. Zarzycki, LSTM and GRU type recurrent neural networks in model predictive control: A Review, Neurocomputing 632 (2025) 129712. https://doi.org/10.1016/j.neucom.2025.129712.
[67] S. Zhao, Z. Xu, Z. Zhu, X. Liang, Z. Zhang, R. Jiang, Short and long-term renewable electricity demand forecasting based on CNN-Bi-GRU model, ICCK Transactions on Emerging Topics in Artificial Intelligence 2 (2025) 1–15. https://doi.org/10.62762/TETAI.2024.532253.

[68] A. Farhadi, A. Zamanifar, A. Alipour, A. Taheri, M. Asadolahi, A hybrid LSTM-GRU model for stock price prediction, IEEE Access 13 (2025) 117594–117618. https://doi.org/10.1109/ACCESS.2025.3586558.
[69] R.M. Farsani, E. Pazouki, A transformer self-attention model for time series forecasting, Journal of Electrical and Computer Engineering Innovations 9 (2021) 1–10. https://doi.org/10.22061/JECEI.2020.7426.391.
[70] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021: pp. 11106–11115. https://doi.org/https://doi.org/10.1609/aaai.v35i12.17325.
[71] D.P. Kingma, M. Welling, Auto-encoding variational bayes, ArXiv Preprint (2013). https://doi.org/10.61603/ceas.v2i1.33.
[72] S. Bond-Taylor, A. Leach, Y. Long, C.G. Willcocks, Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2022) 7327–7347. https://doi.org/10.1109/TPAMI.2021.3116668.
[73] B. Cai, S. Yang, L. Gao, Y. Xiang, Hybrid variational autoencoder for time series forecasting, Knowledge-Based Systems 281 (2023) 111079. https://doi.org/10.1016/j.knosys.2023.111079.

Appendix

A. Lifelong update method

Algorithm 1 Pseudo code for lifelong update method

```
Input: Initial digital twin model structure $DT$, system response datasets $\{\boldsymbol{D}_0, \boldsymbol{D}_1, \ldots\}$ during lifecycle, the empty parameter database $\boldsymbol{CD}$, warm-up phase length $m$, autoencoder structure AE, LSTM structure, LSTM window size $w$
Output: Updated digital twin models at unseen stages
1   $(DT_0, \boldsymbol{\theta}_0) = \text{Train}(DT, \boldsymbol{D}_0)$                # Initialize the digital twin model
2   $\boldsymbol{CD} = \text{store}(\boldsymbol{\theta}_0)$
3   For $i$ in $[1, \ldots, m]$
4       $DT_i, \boldsymbol{\theta}_i = \text{FineTune}(DT_{i-1}, \boldsymbol{D}_i)$        # Fine tune the previous digital twin model
5       $\boldsymbol{CD} = \text{store}(\boldsymbol{\theta}_i)$
6   End
7   For $j$ in $[m+1, \ldots]$                                  # Repeat until the end of lifecycle
        ## Train the AE-LSTM model
8       Encoder, Decoder = Train(AE, $\boldsymbol{CD}$)  # Train autoencoder from stored parameters as Section 2.3.1
9       Latent features $[\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{j-1}]$ = Encoder($\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{j-1}$)   # Obtain the low-dimensional features
10      Training dataset $\boldsymbol{D}_{LSTM} = []$ for LSTM
11      For $i = [w, \ldots, j-1]$
12          $\boldsymbol{x} = [\boldsymbol{s}_{i-w}, \ldots, \boldsymbol{s}_{i-1}]$
13          $\boldsymbol{y} = \boldsymbol{s}_i$
14          $\boldsymbol{D}_{LSTM} = store([\boldsymbol{x}, \boldsymbol{y}])$
15      End
16      LSTM = Train(LSTM, $\boldsymbol{D}_{LSTM}$)                 # Train LSTM for latent features as Section 2.3.2
        ## Estimate digital twin models at future unseen stages
17      $[\boldsymbol{s}'_j, \ldots] = \text{Predict}(\text{LSTM}, \boldsymbol{D}_{LSTM})$     # Predict latent features at future unseen stages as Fig. 5
18      Combined feature dataset $\boldsymbol{FD} = [\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{j-1}, \boldsymbol{s}'_j, \ldots]$
19      For $t$ in $[j, j+1, \ldots]$
20          $\boldsymbol{\theta}'_t = Decoder(\boldsymbol{FD}[t])$        # Predict the model parameters based on estimated features
21          $DT'_t = Deploy(DT, \boldsymbol{\theta}'_t)$             # Deploy the estimated parameters to the model structure
22      End
        ## Refine tune the previous digital twin model based on dataset at current degradation stage
```

```
23        $DT_j, \boldsymbol{\theta}_j = \text{FineTune}(DT_{j-1}, \boldsymbol{D}_j)$
24        $\boldsymbol{CD} = \text{store}(\boldsymbol{\theta}_j)$
25    End
```

Algorithm 1 summarizes the complete workflow of the proposed lifelong DT update framework. The digital twin is first initialized and fine-tuned during a warm-up phase using observed system data. The stored model parameters are then encoded into a latent space, whose temporal evolution is learned by an LSTM and used to predict future parameter features. These predicted latent features are decoded to generate DT models at unseen degradation stages, followed by conventional fine-tuning using newly observed data. This process is repeated throughout the whole lifecycle.

B. Computation Efficiency

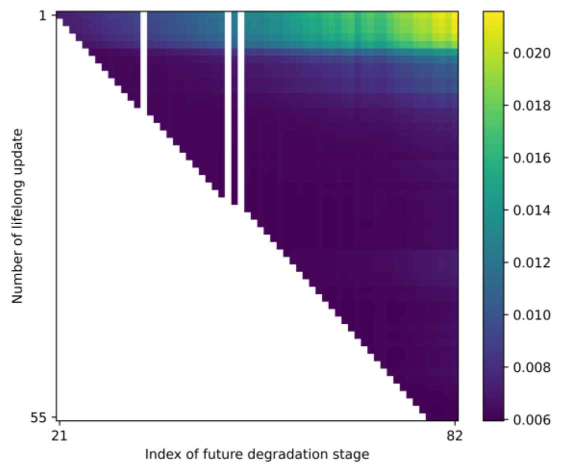
All tests were conducted on a low-cost desktop (Intel Core i7-3770 CPU@3.40GHz processor, 24.0 GB RAM). The average CPU times for each update step across all test problems are summarized in Table 3. The CPU time required to fine-tune the DT model remains relatively stable, with average values of approximately 0.038 seconds for the battery cases and 0.05 seconds for the engine pressure cases. In contrast, training the AE-LSTM model requires substantially longer computational time, around 20 seconds for battery cases and 47 seconds for engine pressure cases. The time difference is because a deeper DT model was adopted for engine pressure cases, which results in a deeper AE-LSTM model. As this study performs the lifelong update after each system lifecycle, the required CPU time is negligible compared to the cycle duration, which are 2 hours per charging-discharging cycle in the battery dataset and 3.3 hours per flight cycle in the engine pressure dataset.

Table 3. Average and standard deviation of CPU time required for each update step.

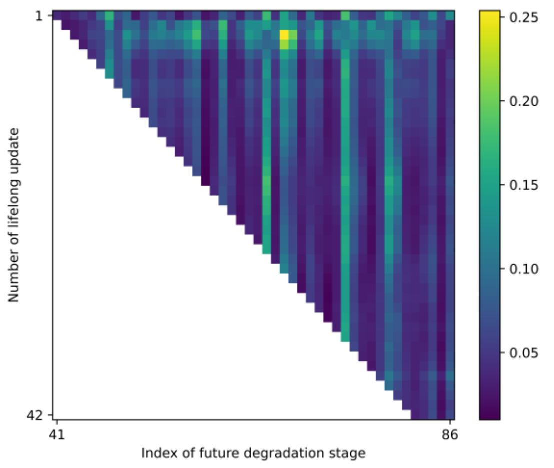
Test problem	Time to fine tune the DT model	Time to train AE-LSTM
Battery charging	0.0378 ± 0.0120 s	20.5257 ± 0.7359 s
Battery discharging	0.0382 ± 0.0076 s	20.6285 ± 0.7728 s
Engine pressure p_1	0.0513 ± 0.0117 s	47.1125 ± 2.2488 s
Engine pressure p_2	0.0476 ± 0.0045 s	47.2332 ± 1.3877 s

C. Variance analysis

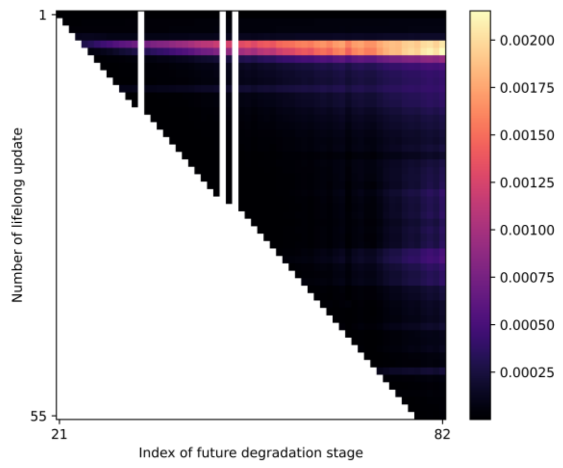
This section explores the two variance sources, i.e., the initial training of the DT model and the subsequent update procedures. To isolate these effects, two comparison studies are performed with controllable random seeds.



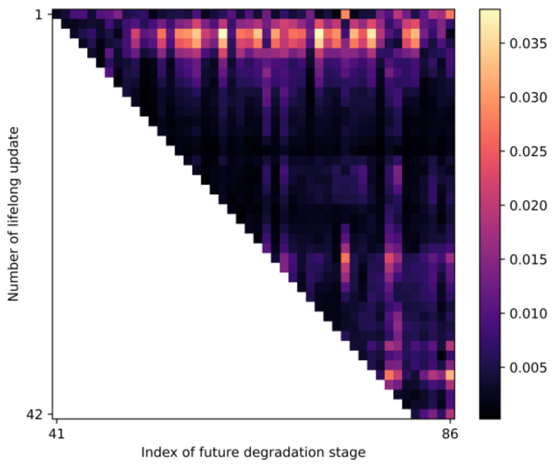
(a) Battery discharging, mean MSE



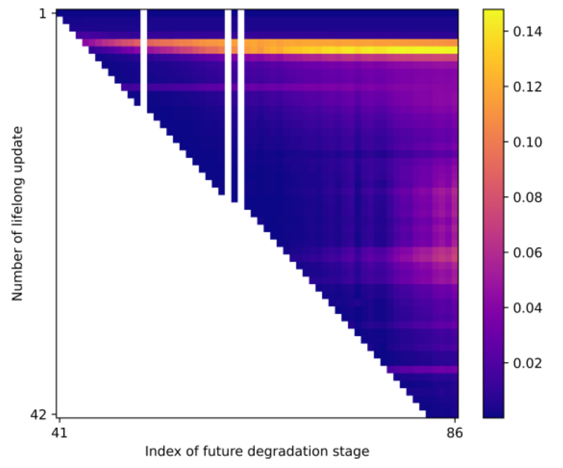
(d) Engine pressure p_1, mean MSE



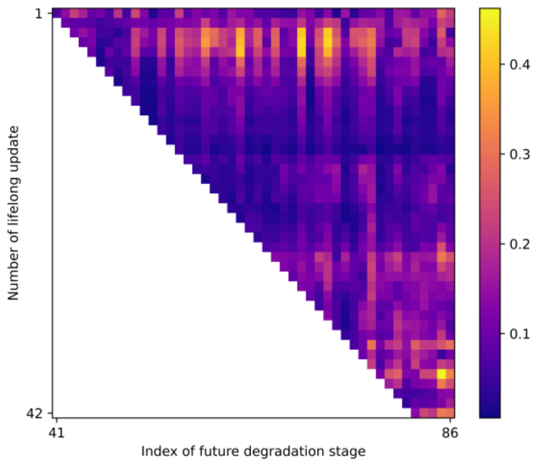
(b) Battery discharging, STD of MSE



(e) Engine pressure p_1, STD of MSE



(c) Battery discharging, STD/mean of MSE



(f) Engine pressure p_1, STD/mean of MSE

Fig. 15. Variance study of the proposed lifelong update method

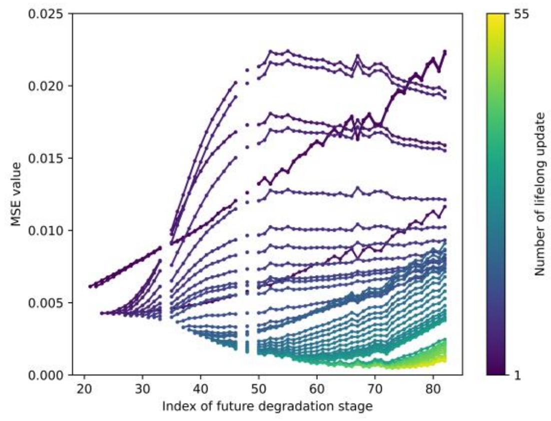
In the first comparison study, the initial DT model is consistently initialized and trained using a fixed random seed (i.e., 10^2). Both the fine-tuning and the proposed lifelong update methods are executed using the same random seed per run, specifically $\{10, 10^2, 10^3, 10^4, 10^5\}$, to assess the effect of stochasticity in the update process. As in Sections 3 and 4, each run provides a group of MSE curves. From these, the mean and standard deviations (STD) from MSE values at each future degradation stage after completing one update are computed across the five runs to quantify the inherent randomness of the lifelong update method.

Fig. 15 summarizes the mean MSE value, the STD values of MSE, and the ratios of STD to mean MSE at each degradation stage for battery discharging and engine pressure cases. Although some large errors exist at former lifelong updates in both problems (e.g., mean MSE values over 0.020 in Fig. 15 (a), several MSE values over 0.25), the STD values of MSE remain low across future degradation stages, as shown in Fig. 15 (b) and (e). Specifically, the STD values at most degradation stages are smaller than 0.0005 and 0.010 for battery discharging and engine pressure, respectively. Most STD-to-mean ratios are under 0.02 in battery discharging problem, and below 0.10 in the engine pressure p_1 problem. Meanwhile, results in the shared data repository show that the proposed lifelong update method outperforms the conventional fine-tuning method and shows similar performance consistently across five independent runs. One potential reason is that the proposed model structure does not involve additional stochastic operations (e.g., dropout). The training process would converge to a stable range after completing 1000 epochs. Therefore, given an initial DT model, the proposed lifelong update method demonstrates a degree of robustness.

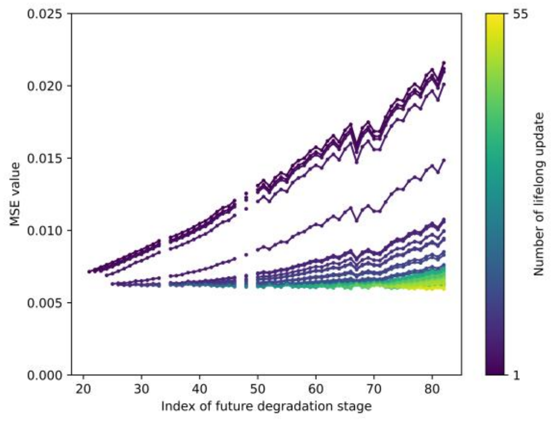
In the second comparison study, different random seeds are used to train the initial DT model, while fixing the seed for the update process. Results in Fig. 16 demonstrates that the initial DT model could lead to noticeably different performance convergence of the update DT. For instance, the MSE values for battery discharging could converge around 0.001 when the random seed 10 (Fig. 16 (a)), while they converge around 0.005 when the random seed is 100 (Fig. 16 (b)). Similar behavior is observed in the engine pressure results in Fig. 16 (c) and Fig. 16 (d).

This confirms that the initialization of the DT model is the dominant source of variance.

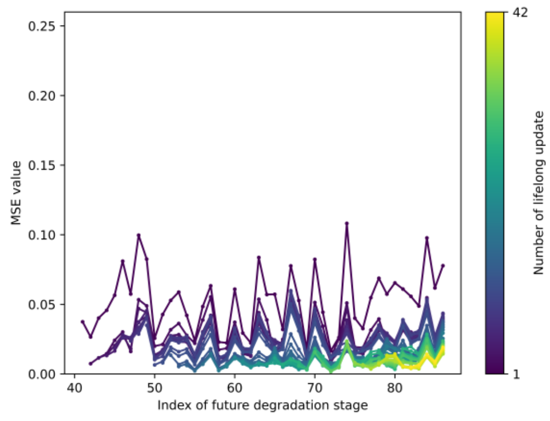
Based on the above analysis, the proposed lifelong update method is robust given a fixed initial DT model. Therefore, the results in Section 3, Section 4, and the subsequent discussion are reported using a representative run where the random seed 100 is used for both the DT model construction and update methods. Additional results and detailed comparisons can be found in the shared repository.



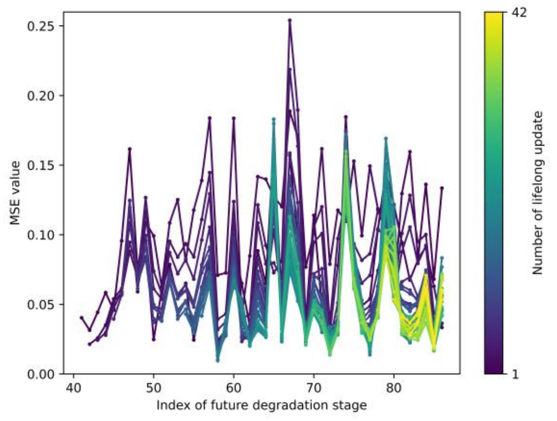
(a) Battery discharging, 10-10



(b) Battery discharging, 100-10



(c) Engine pressure p_1, 10-10



(d) Engine pressure p_1, 100-10

Fig. 16. Variance study of initial DT model training. Random seeds a-b: a for the initial DT model, and b for the update method.

D. Time-series forecasting method

This section explores the potential of other state-of-the-art model structures, e.g., Transformer [64] and Gated Recurrent Unit (GRU) [65], for the proposed lifelong update task.

The GRU is a simplified variant of LSTM via merging the forget and input gates into a single update gate and removing the output gate, resulting in only two gates: update and reset. This streamlined structure reduces the number of parameters and generally leads to faster training while maintaining comparable performance, which has been applied to model predictive control [66], energy prediction [67], and stock prediction [68]. Different from LSTM and GRU relying on the recurrence structure and the gating mechanism to process the data sequentially, Transformer adopts the attention mechanism to process all input tokens (i.e., data in the time-series sequence) simultaneously, where the importance of different inputs relative to each other is weighted in parallel. The most relevant parts of the input sequences are then identified, ensuring that the Transformer can capture both short-term fluctuations and long-term trends in complex datasets, such as fuel cell degradation [33], energy consumption [69], and weather conditions [70].

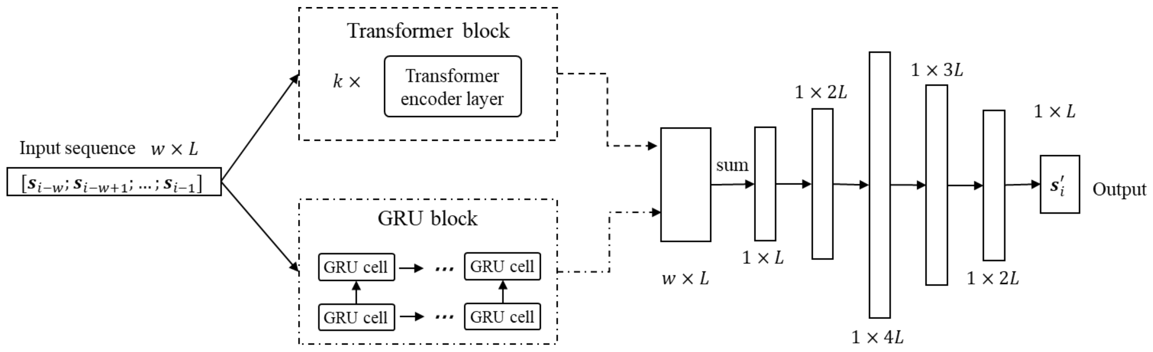


Fig. 17. Transformer and GRU structures.

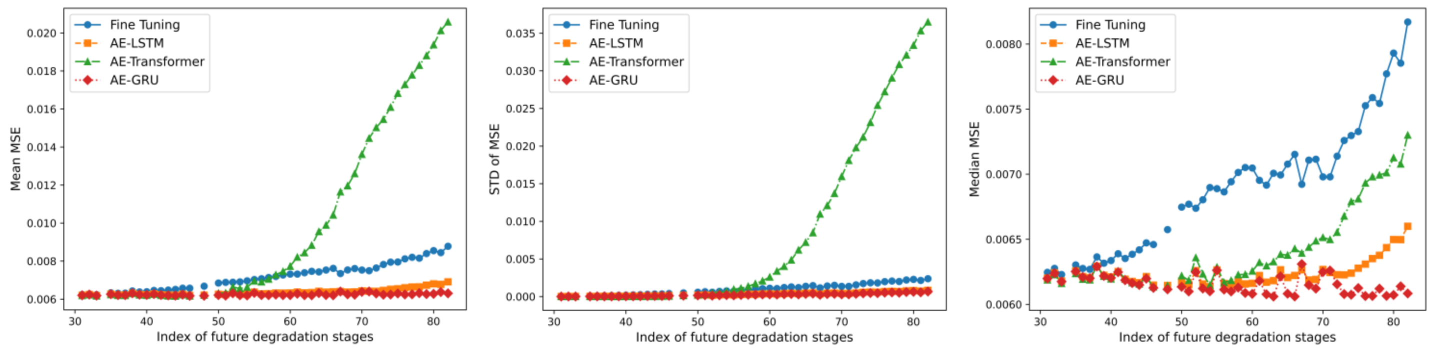
The structures of the implemented Transformer in the study are shown in Fig. 17. Compared with the LSTM model in Section 2.3.2, the only difference is replacing the 2-layer LSTM block as the Transformer block to capture the encoded features from the input sequence $[\boldsymbol{s}_{i-w}; \dots; \boldsymbol{s}_{i-1}]$ obtained during the stage window $[i - w, i - 1]$. The applied encoder block includes k transformer encoder layers, where each layer has a multi-head attention part and a feedforward model. The encoder block is implemented by the Pytorch library in the work. All $k = 6$ encoder layers share the same hyperparameter setting, such as $2L$ as the dimension of the feed-forward network model, 1 as the number of attention heads, and 0 as the dropout rate. The output dimension of the encoder block is the same as the input dimension. A sum operator is then applied to convert the encoder output matrix to the encoded feature vector, whose size is $1 \times L$. Finally, several hidden layers with the activation function Tanh$(\cdot)$

are assigned to obtain the final output as the predicted latent feature vector $\boldsymbol{s}_i'$ at the next i-th stage.

The tested GRU model shares the same structure as LSTM, while the LSTM cell is replaced by the GRU cell in the 2-layer GRU block, as shown in Fig. 17. The hidden size and the dropout rate are identical to those of the LSTM model. The output of the last cell in the second GRU layer is then feedforward to the remaining hidden layers to obtain the final output as the predicted latent feature vector $\boldsymbol{s}_i'$ with the size $1 \times L$.

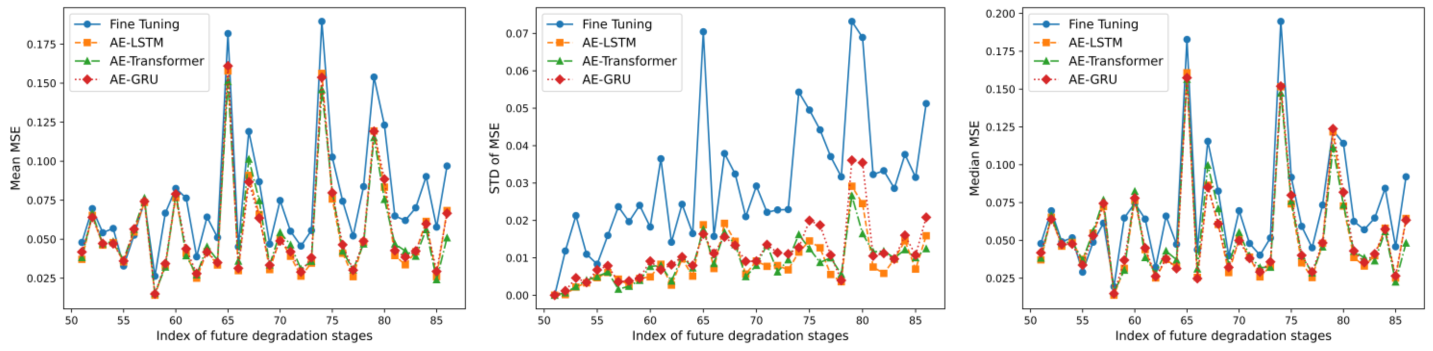
Both implemented Transformer and GRU models are integrated into the proposed autoencoder and evaluated using the battery discharging and the engine pressure p_1 datasets. For clarity, the two models are denoted as AE-Transformer and AE-GRU, respectively, in the subsequent discussion. The training and testing settings for both models are identical to those used for AE-LTM in Section 3 and Section 4. After completing one lifelong update with AE-Transformer or AE-GRU, the MSE values of updated DTs are compared across all future degradation stages, as shown in Fig. 18 and Fig. 19.

The results for the battery discharging case show that AE-LSTM and AE-GRU outperform the conventional fine-tuning method significantly with smaller mean, STD, and median of MSE values, as shown in Fig. 18 (a), Fig. 18 (b), and Fig. 18 (c) respectively. Although AE-GRU achieves better median and mean MSE values than AE-LSTM, their STD values are close, indicating similar robustness. The AE-Transformer also achieves smaller median MSE values than the fine-tuning method. However, the corresponding mean and STD of MSE values are lager than other methods after the 60th degradation stage, indicating that AE-Transformer has a worse performance robustness in this problem.

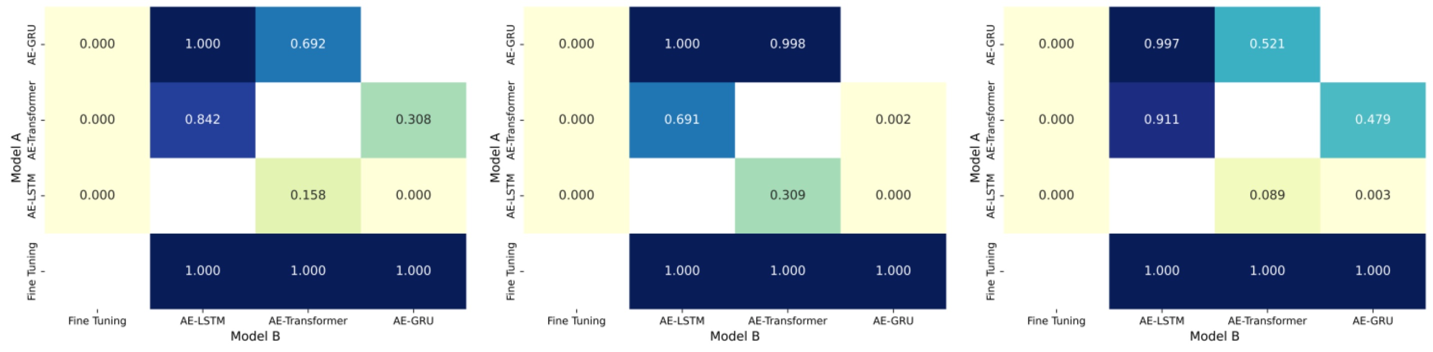


(a) Mean MSE (b) STD of MSE (c) Median MSE

Fig. 18. Comparison of different time-series models in battery discharging.



(a) Mean MSE (b) STD of MSE (c) Median MSE



(d) One-tailed *t*-test based on mean MSE (e) One-tailed *t*-test based on STD of MSE (f) One-tailed *t*-test based on median MSE

Fig. 19. Comparison of different time-series models on engine pressure p_1.

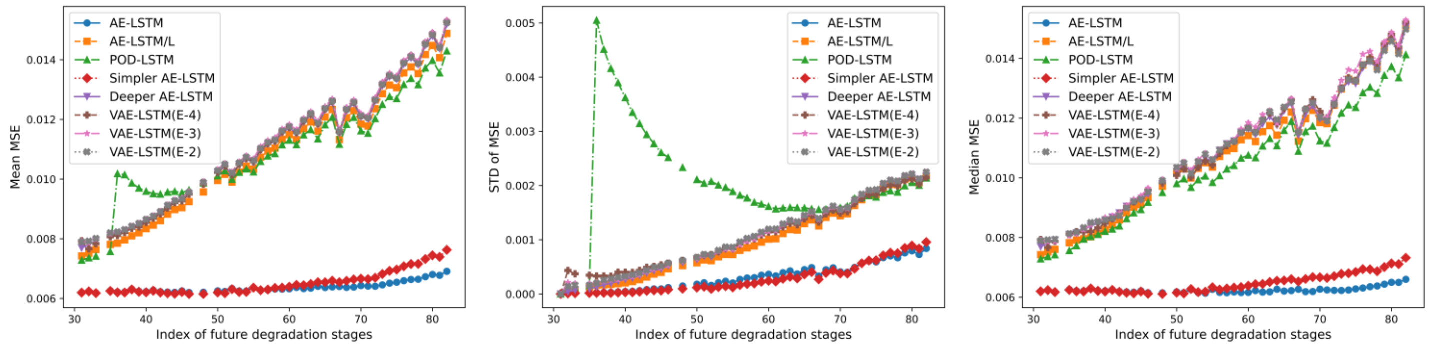
The results for engine pressure demonstrate that the mean (Fig. 19 (a)), STD (Fig. 19 (b)), and median (Fig. 19 (c)) of MSE values obtained by AE-LSTM, AE-Transformer, and AE-GRU are consistently smaller than those produced by the conventional fine-tuning method at each degradation stage. To further examine the statistical significance of performance improvement, the one-tailed *t*-test is conducted for each model pair (A, B) to infer whether model A has significantly smaller MSE values than model B. Corresponding p-values are summarized from Fig. 19 (d) to Fig. 19 (f). In general, the tested three models achieve significantly better performance than the fine-tuning method, as the corresponding p-values are all 0.000 (<0.05). Meanwhile, AE-LSTM achieves significantly smaller mean, STD, and median of MSE values than AE-GRU, as the p-values are 0.000 in Fig. 19 (d), 0.000 in Fig. 19 (e), and 0.003 in Fig. 19 (f) respectively. In contrast, no significant performance improvement exists in other model pairs in most cases. For the mean MSE in Fig. 19 (d), the p-value is 0.158 for model pair (AE-LSTM, AE-Transformer), 0.842 for model pair (AE-Transformer, AE-LSTM), 0.308 for model pair (AE-Transformer, AE-GRU), and 0.692 for model pair (AE-GRU, AE-Transformer). As those values are larger than 0.05, no significant performance improvements are observed among those model pairs. The similar behavior also exists in STD and median of MSE values, as shown in Fig. 19 (e) and Fig. 19 (f) respectively. Therefore, the three time-series models have comparable performance in this problem, except that LSTM outperforms GRU.

These findings confirm that the proposed lifelong update method is applicable to be integrated with different time-series forecasting models, which would provide comparable performance and outperform the conventional fine-tuning method in terms of performance and robustness for future unseen degradation stages.

E. Feature extraction methods

The proposed method adopts the autoencoder as the nonlinear feature extraction method to capture the latent features of DT models. Some comparison studies are performed to study the feature extraction methodology, the model complexity, and the variational autoencoder on battery discharging and engine pressure p_1 problems.

During tests, AE-LTSM outperforms all compared variants on the battery discharging problem significantly, in terms of smaller mean, STD, and median of MSE values at each degradation stage, except that AE-LSTM and simpler AE-LSTM have comparable STD of MSE values, indicating similar robustness, as shown in Fig. 20. In the remaining of this section, only results on the engine pressure p_1 problem are presented with the definitions of compared variants.



(a) Mean MSE (b) STD of MSE (c) Median MSE

Fig. 20. Comparison results of the feature extraction method in the battery discharging problem.

E1. Feature extraction methodology

In this work, parameters of all hidden layers in the FNN model are processed by an autoencoder together to capture potential dependencies among hidden layers. To study the effect of the applied option, this test conducts feature extraction for each hidden layer independently. For an FNN model with N hidden layers, N autoencoder models are trained to reconstruct the parameters of each layer, respectively. Then N LSTM models are adopted to capture the changing behavior of learned parameter features during the lifecycle for each hidden layer. During the test, the autoencoder structure of the i-th layer is shown as Fig. 21, where the output of each hidden layer is activated by Tanh(·). The LSTM model adopted in the test is the same as that in Section 2.3.2. The training and testing settings are identical to those in Section 3 and Section 4. To clarify the following discussion, the tested method in this section is denoted as Autoencoder-LSTM per layer (AE-LSTM/L).

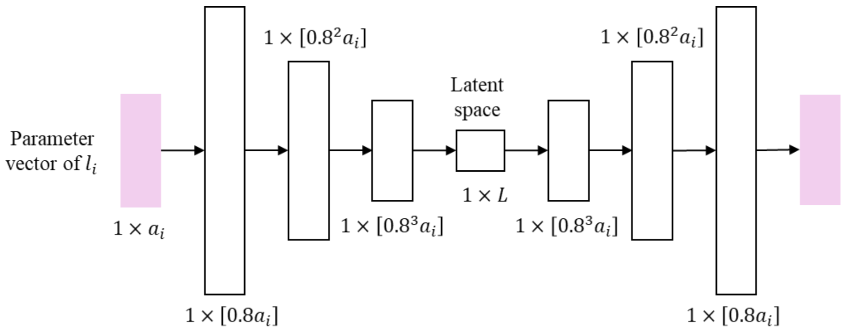


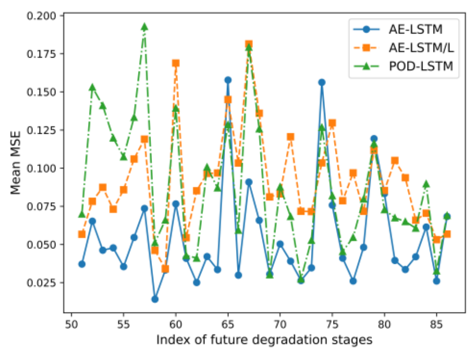
Fig. 21. Autoencoder structure for a single hidden layer. a_i is the number of parameters in the hidden layer l_i. $[\cdot]$ refers to the nearest integer. 0.8 is defined manually to reduce the number of hidden nodes by 80%.

Meanwhile, the autoencoder is a nonlinear feature extraction method, whose performances are compared with a linear feature extraction method, proper orthogonal decomposition (POD), also known as principal component analysis. In this test, the autoencoder is replaced by POD to extract latent features from DT model parameters, which are then used to train the LSTM model to capture their temporal evolution. For clarification, the tested method is defined as POD-LSTM.

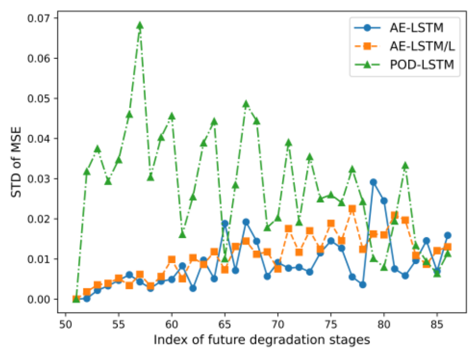
Comparison results on the engine pressure p_1 are summarized in Fig. 22. Compared with AE-LSTM/L and

POD-LSTM, AE-LSTM provides smaller mean and media of MSE values at each degradation stage as shown in Fig. 22 (a) and Fig. 22 (c), whose corresponding p-values are all 0.000 in Fig. 22 (d) and Fig. 22 (f). For the STD of MSE values, AE-LSTM has smaller values in most stages as shown in Fig. 22 (b). The corresponding p-values are 0.041 for model pair (AE-LSTM, AE-LSTM/L) and 0.000 for model pair (AE-LSTM, POD-LSTM). Those results show that AE-LSTM outperforms both AE-LSTM/L and POD-LSTM with sufficient statistical significance.

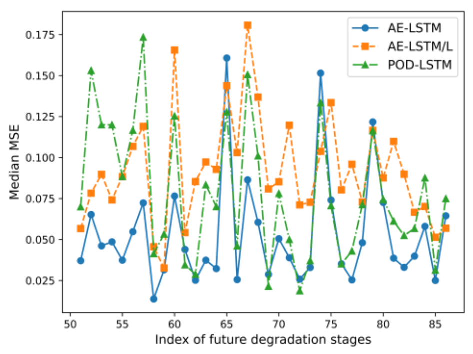
The above discussions demonstrate that feeding the parameters of all hidden layers to the autoencoder simultaneously would benefit learning latent features that consider dependencies among hidden layers. This integrated approach improves the accuracy and robustness of the final reconstructed DT models, demonstrating its advantage over the layer-wise alternative. Meanwhile, nonlinear feature extraction methods outperform the linear ones in capturing the latent features of the DT model parameters. This performance gap is attributed to the inherently nonlinear nature of parameter evolution in DT models, which cannot be effectively captured using linear techniques.



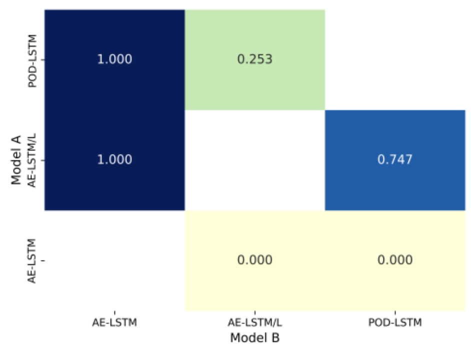
(a) Mean MSE

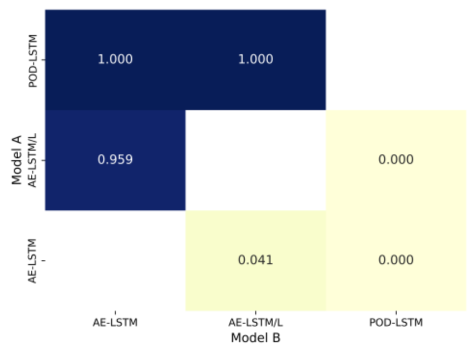


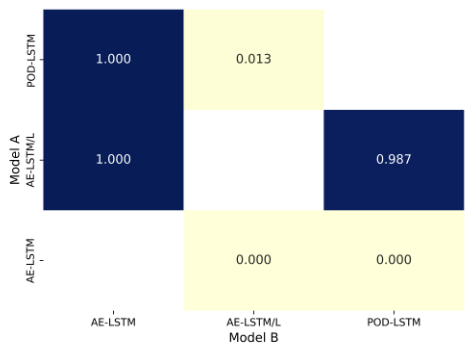
(b) STD of MSE



(c) Median MSE







(d) One-tailed t-test based on mean MSE (e) One-tailed t-test based on STD of MSE (f) One-tailed t-test based on median MSE

Fig. 22. Comparison of different feature extraction methodologies on engine pressure p_1.

E2. Model complexity

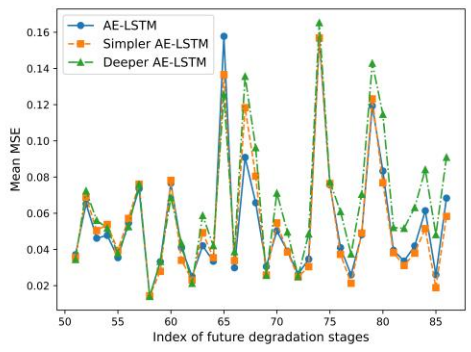
In this paper, the structure of the nonlinear feature extraction method is determined through trial-and-error. Further investigation is conducted to evaluate the impact of different autoencoder complexities on the performance of the updated DT models. Specifically, two distinct Autoencoder structures with details in Table 4 are compared with the baseline configuration proposed in Fig. 3.

Generally, the simpler and deeper autoencoder models follow the same overall structure in Fig. 3 but differ in the number of hidden layers and the dimension of the preprocessed parameter vectors. In the simpler Autoencoder, the parameter vector of each hidden layer in the DT model is first reduced to a 16-dimensional vector. These reduced vectors from all hidden layers are then concatenated into a single larger vector, which is subsequently processed by an encoder with two hidden layers to extract the latent features of the model parameters. During the decoder part, two hidden layers are adopted to reconstruct the model parameters based on the latent features. In contrast, the deeper Autoencoder expands the preprocessing dimension to 256 and employs six hidden layers in both the encoder and decoder.

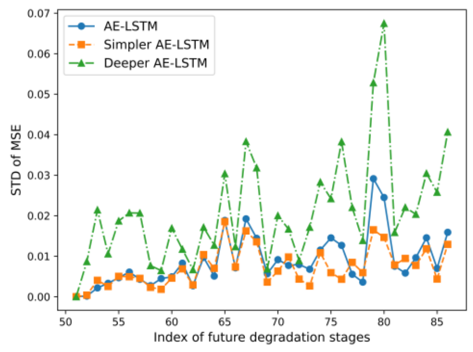
Table 4. Different autoencoder structures for comparison

Autoencoder	Encoder	Decoder
Simpler	$16N \rightarrow 8N \rightarrow L$	$L \rightarrow 8N \rightarrow 16N$
Baseline	$64N \rightarrow 32N \rightarrow 16N \rightarrow 8N \rightarrow L$	$L \rightarrow 8N \rightarrow 16N \rightarrow 32N \rightarrow 64N$
Deeper	$256N \rightarrow 128N \rightarrow 64N \rightarrow 32N \rightarrow 16N \rightarrow 8N \rightarrow L$	$L \rightarrow 8N \rightarrow 16N \rightarrow 32N \rightarrow 64N \rightarrow 128N \rightarrow 256N$

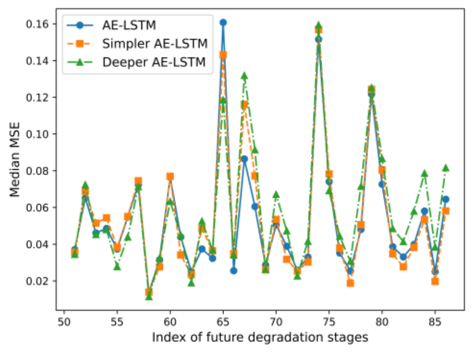
Results on the engine pressure p_1 are summarized in Fig. 23. Generally, AE-LSTM outperforms the deeper one significantly in all performance metrics as shown from Fig. 23 (a) to Fig. 23 (c). The corresponding p-values are 0.000 for both mean and STD of MSE values, and 0.014 for the median MSE values, indicating significant performance improvement. In contrast, AE-LSTM is comparable to the simpler one in this case. The p-value for the model pair (simpler AE-LSTM, AE-LSTM) is 0.006 for the STD of MSE values, which shows the AE-LSTM has a statistically higher robustness. The p-values on both their mean and median of MSE values, however, are larger than 0.05 as shown in Fig. 23 (d) and Fig. 23 (f), indicating no significant differences between the two model structures from these two metrics.



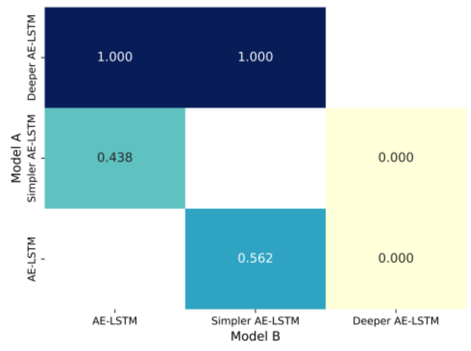
(a) Mean MSE



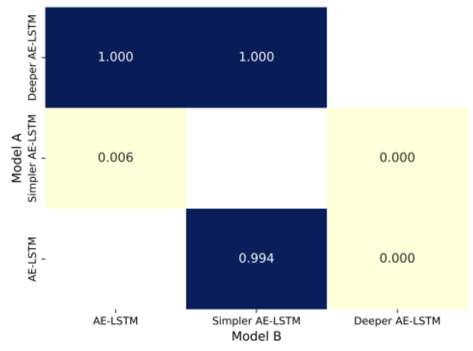
(b) STD of MSE



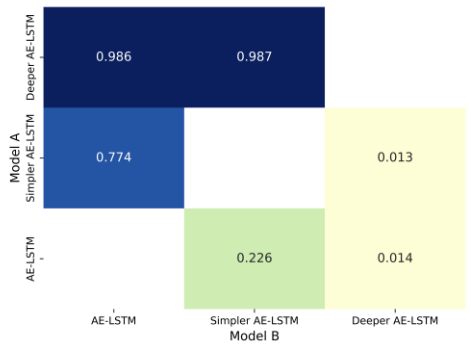
(c) Median MSE



(d) One-tailed t-test based on mean MSE



(e) One-tailed t-test based on STD of MSE



(f) One-tailed t-test based on median MSE

Fig. 23. Comparison of AE-LSTM with different model depths on engine pressure p_1.

Combining results on both the battery discharging and engine pressure cases, the proposed AE-LSTM model provides more robust and better performance than the simpler and deeper counterparts. The simpler model may suffer from underfitting, as its limited capacity fails to capture complex patterns in the training data. Conversely, the deeper model appears to overfit, resulting in poor generalization to unseen degradation stages.

E3. Variational autoencoder

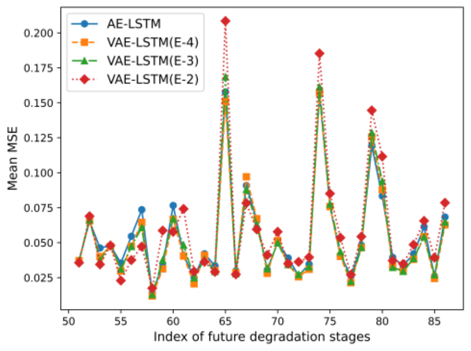
Although an autoencoder is a powerful tool for data reconstruction and dimensionality reduction, it often suffers from an unstable and uninterpretable latent space. This limitation has been solved by the variational autoencoder (VAE) [71]. Different from autoencoder mapping input to a deterministic latent representation, VAEs learn a stochastic latent space characterized by a Gaussian distribution with learnable mean and variance. During training, the Kullback–Leibler (KL) divergence between the learned latent distribution and the standard normal distribution is minimized as a regularization term, preventing the latent space from becoming irregular or overfit. This probabilistic formulation enables the model to explore a smooth and continuous latent space, facilitating more robust representations and improving its ability to tackle a variety of generative modeling tasks [72]. Moreover, due to its improved generalization capability, VAE has also been applied in time-series forecasting, where it facilitates latent dynamics modeling and supports uncertainty-aware prediction [73].

To further explore the potential of the proposed update framework, the performance of VAE is investigated. In the study, the VAE adopts the same network structure as the autoencoder model, with the addition of an extra hidden layer to learn the variance in the latent space. The original loss function in Eq. (4) is extended by incorporating a weighted KL divergence term as a regularization component. As this work focuses on forecasting rather than generative modeling, only the learned mean of the latent distribution in VAE is used to train the LSTM model to capture the temporal evolution of the DT model parameters. All other training hyperparameters are identical to those in Sections 3.2 and 4.2 for both problems. The tested method is defined as VAE-LSTM for clarification in the following discussion.

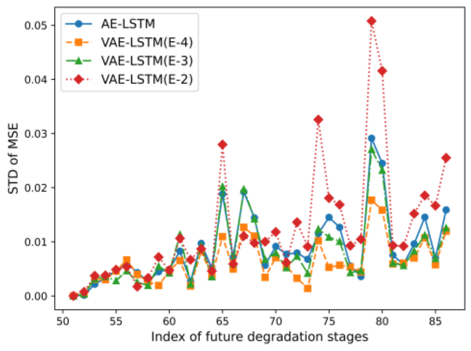
The performances of AE-LSTM and three VAE-LSTM models with different weights (10^{-4}, 10^{-3}, and 10^{-2}) of KL divergence are summarized in Fig. 24. Overall, the performance of VAE-LSTM decreases with larger KL divergence weights. In Fig. 24 (d), the p-values for model pairs (VAE-LSTM(E-4), VAE-LSTM(E-3)), (VAE-LSTM(E-4), VAE-LSTM(E-2)), and (VAE-LSTM(E-3), VAE-LSTM(E-2)) are 0.039, 0.003, and 0.001, respectively, for mean MSE values, reflecting a better performance with smaller weights. A better performance robustness is also observed in models with smaller KL divergence weights, as shown in Fig. 24 (e), where all corresponding p-values are zero. Although median MSE values are comparable between VAE-LSTM(E-4) and VAE-LSTM(E-3), both models outperform VAE-LSTM trained with the weight 10^{-2}, as p-values are 0.006 and 0.003, respectively, shown in Fig. 24 (f). Besides, only VAE-LSTM(E-4) achieves better performance than AE-

LSTM, considering that the corresponding p-values are 0.000 in mean (Fig. 24 (d)) and STD (Fig. 24 (e)) of MSE values, and 0.041 in the median MSE values (Fig. 24 (f)). The VAE-LSTM model trained with larger weights are comparable or even worse than AE-LSTM. For instance, the p-values for model pair (AE-LSTM, VAE-LSTM(E-4)) are smaller than 0.05 in three performance metrics, while the p-values for model pair (AE-LSTM, VAE-LSTM(E-3)) are larger than 0.05 in most cases.

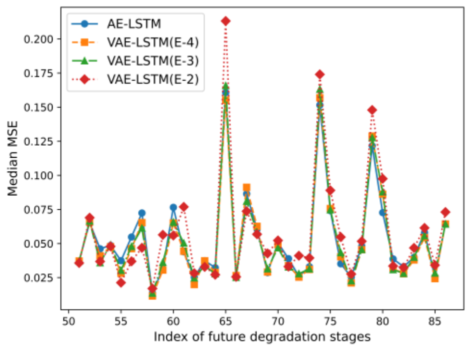
Although VAE-LSTM shows inferior performance on the battery discharging, the above results on engine pressure demonstrate that integrating VAE into the proposed framework is effective, provided that a suitably small KL divergence weight is chosen to balance latent regularization and reconstruction accuracy.



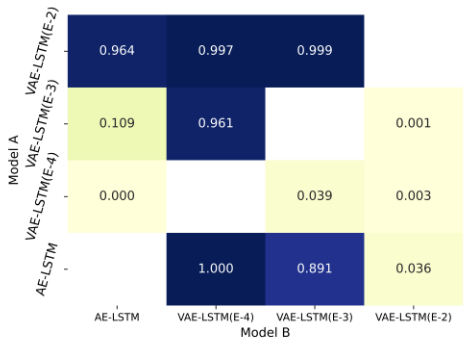
(a) Mean MSE



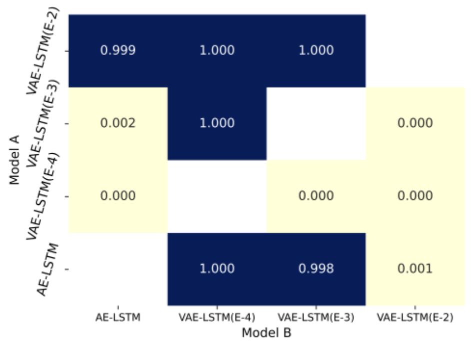
(b) STD of MSE



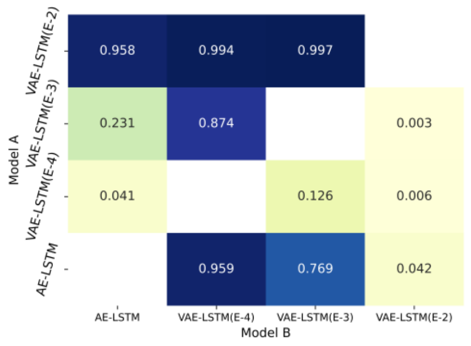
(c) Median MSE



(d) One-tailed t-test based on mean MSE



(e) One-tailed t-test based on STD of MSE



(f) One-tailed t-test based on median MSE

Fig. 24. Comparison of AE-LSTM and VAE-LSTM on engine pressure p_1.

F. Hyperparameter analysis

In this section, a hyperparameter sensitivity analysis is performed for the latent feature dimension, the LSTM window size, and the warm-up phase length. All other settings are identical to those in Section 3. The evaluations are then conducted on the battery discharging problem, and the performance is assessed over all future degradation stages during the lifelong update phase.

F1. Latent feature dimension

In the proposed autoencoder model, the entropy of the initial DT model is adopted to determine the latent space dimension. The initial model entropy varies with different β values in Eq. (1). As shown in Fig. 25, the entropy value is approximately 7 when $\beta = 0$ and decreases exponentially towards zero as β increases. To further explore the effect of latent feature dimension, AE-LSTM is trained under a set of different dimensions from 1 to 40, whose performances on all future degradation stages are summarized in Fig. 26. The one-tailed t-test is adopted to infer the statistical performance improvement for dimension A compared with dimension B.

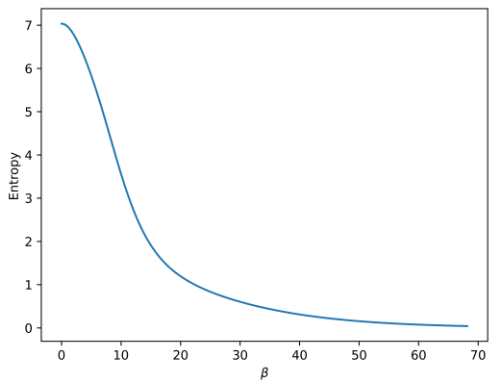
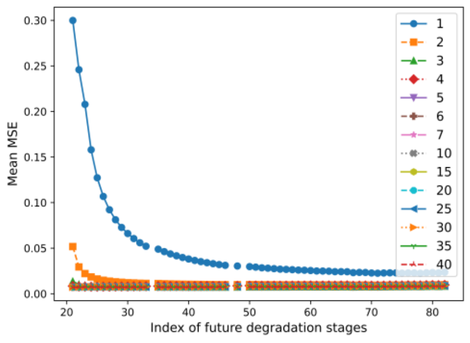
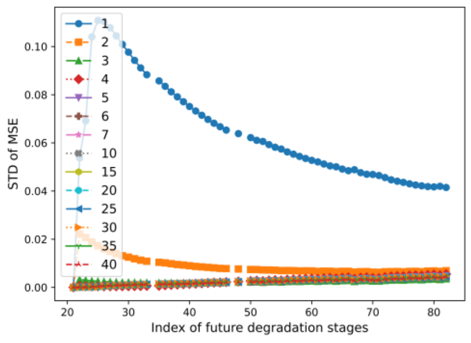


Fig. 25. Entropy of initial DT model parameters over various β values.

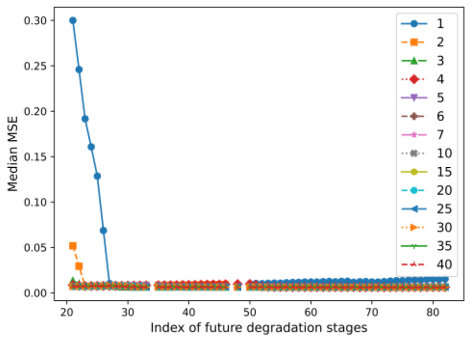
From Fig. 26 (a)-(c), the mean, STD, and median of MSE values decrease as the latent feature dimension increases, indicating a general improvement in AE-LSTM performance with a larger latent space. The one-tailed t-test results in Fig. 26 (d)-(f) further quantify the trend of performance improvements. More specifically, when the latent dimension B is smaller than 6 (except for 3), the p-values in the upper-left triangular region are all close to 0.00. Those values indicate that any larger latent dimension A>B yields significantly smaller MSE. The feature dimension 7 achieves the best performance for AE-LSTM, as the p-values for the dimension pair (7, B) is always smaller than 0.05 for any dimension B. The latent dimension 10 provides the second-best performance, as the p-values for the dimension pair (10, B) are smaller than 0.05 in most cases. For dimensions larger than 20, the p-values in most upper-left entries remain near zero, confirming that these larger latent spaces also outperform most smaller ones. Overall, these results suggest that latent dimensions smaller than the max entropy of the initial model (i.e., 7 when $\beta = 0$ as shown in Fig. 25) cannot capture the degradation behavior in the system lifecycle, resulting in a worse performance, while larger latent dimensions consistently achieve better performance statistically in most cases.



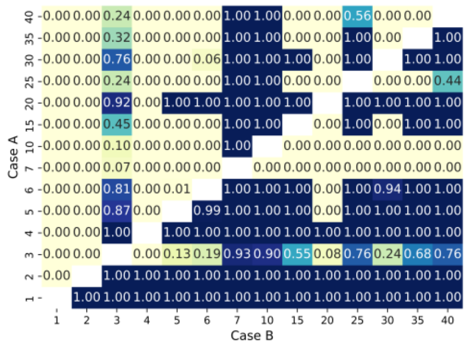
(a) Mean MSE



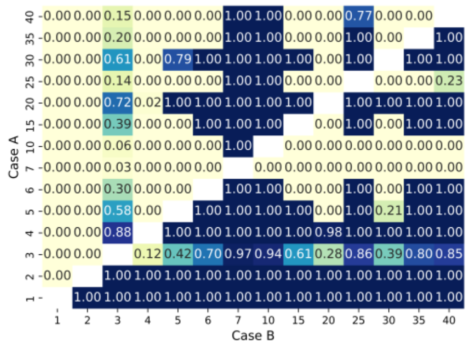
(b) STD of MSE



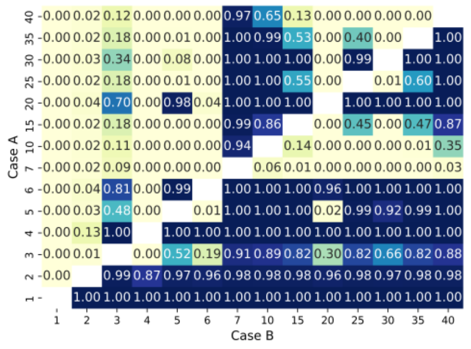
(c) Median MSE



(d) One-tailed *t*-test based on mean MSE



(e) One-tailed *t*-test based on STD of MSE



(f) One-tailed *t*-test based on median MSE

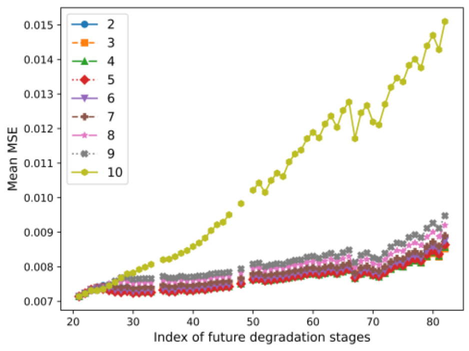
Fig. 26. Comparison of different latent feature dimensions.

These observations demonstrate that a small β value and a moderate value of the scaling factor α in Eq. (3) are effective in achieving informative latent feature expression. However, excessively large values of α can result in high-dimensional latent spaces, increasing model complexity and computational cost during training.

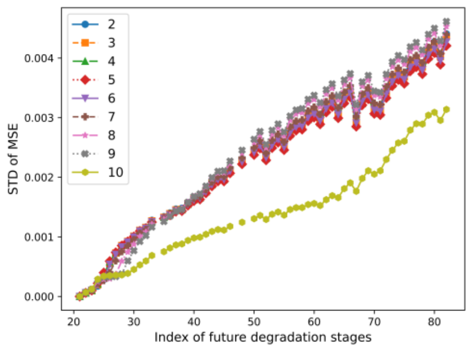
F2. LSTM window size

The window size in the LSTM determines how much historical information is included from previous stages, and thus directly affects the temporal representation quality. The results for different window sizes using the AE-LSTM model are summarized in Fig. 27. Generally, increasing the window size degrades the model performance in the study. As shown in Fig. 27 (a)-(c), the largest window size 10 produces the minimal STD across stages, but also achieves the worst mean and median MSE values.

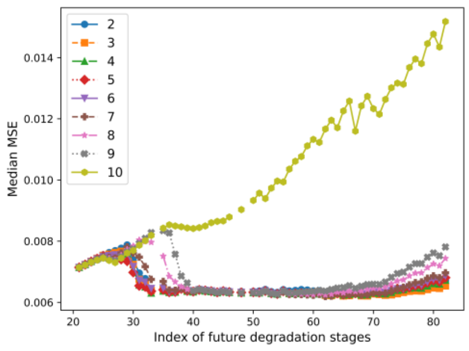
The one-tailed t-test results in Fig. 27 (d)-(f) provide the statistical evidence. The p-values in the lower-right triangular region are almost zero, indicating that a smaller window size would outperform a larger size in the mean, STD, and median of MSE values statistically. More specifically, when the window size A increases from 2 to 4 or 5, the overall performance increases as more p-values smaller than 0.05 are observed in the mean and STD of MSE values. However, the performance decreases further if the window size continuously increases over 6. Therefore, the window size 4 for 5 provides the best trade-off between information coverage and prediction accuracy in the battery discharging problem.



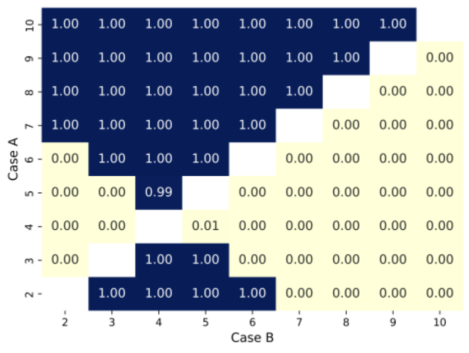
(a) Mean MSE



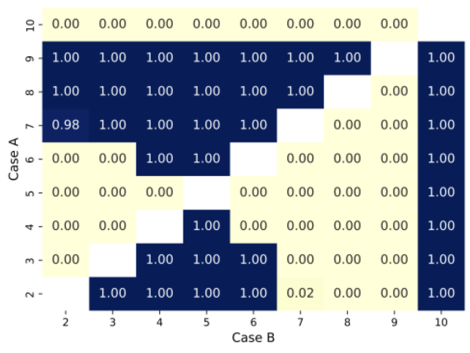
(b) STD of MSE



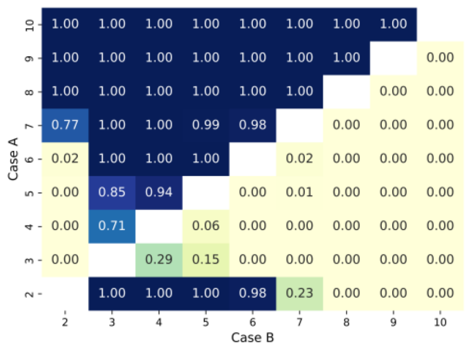
(c) Median MSE



(d) One-tailed t-test based on mean MSE



(e) One-tailed t-test based on STD of MSE



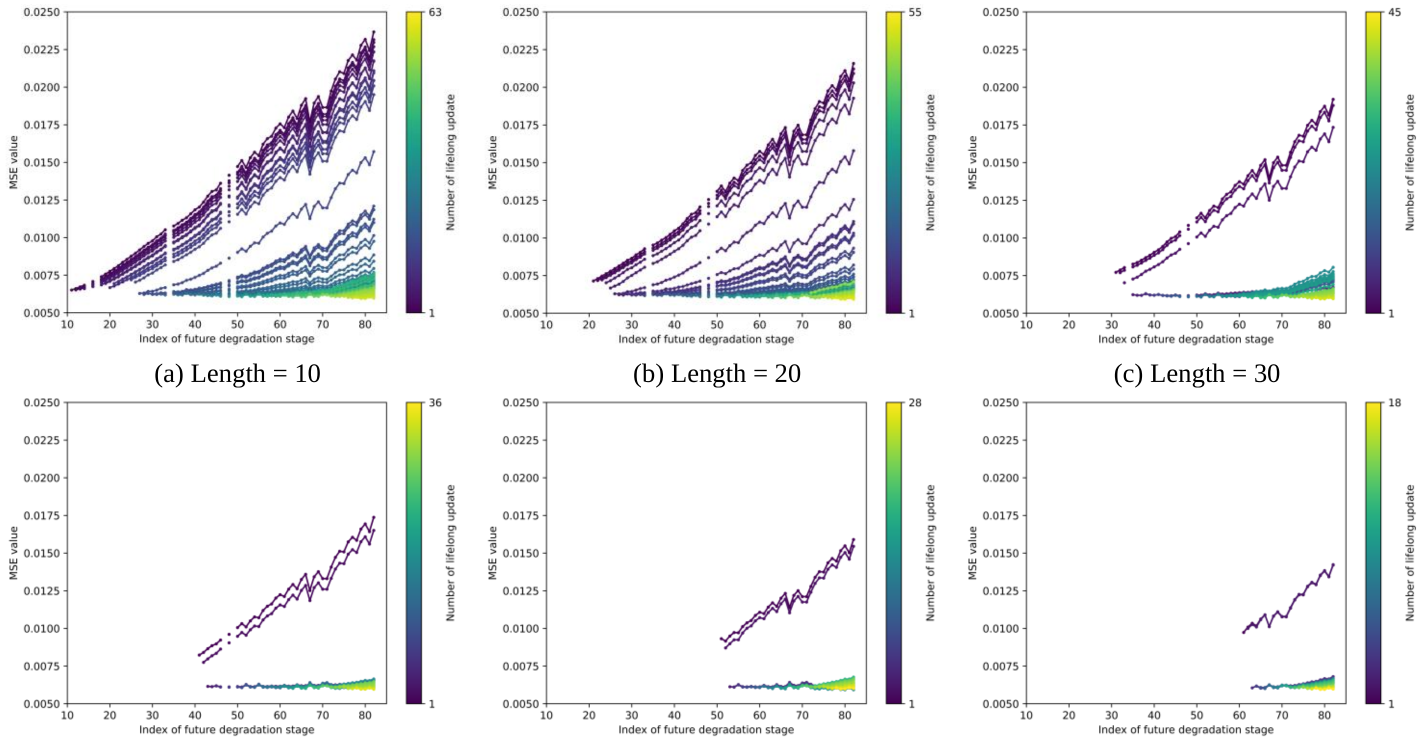
(f) One-tailed t-test based on median MSE

Fig. 27. Comparison of different window sizes for LSTM models.

F3. Warm-up phase length

In the proposed framework, the warm-up length determines how much historical degradation information is available before performing the lifelong update phase. This study compares the performance of updated DT models with different warm-up lengths. The MSE curves at all future degradation stages are summarized in Fig.

28 for each case. Although the performances are relatively poor at the first several lifelong updates immediately after the warm-up phase, increasing the warm-up phase length consistently improves the model performance at later stages. For instance, at the 80th future degradation stage, the number of MSE values over 0.0125 is around 12 when the warm-up phase length is 10. This number decreases sharply with the phase length, e.g., 6 at length 20, 3 at length 30, 2 at lengths 40 and 50, and only 1 at length 60. Meanwhile, the ME value range at the 80th degradation stage reduces as the warm-up phase length increases, indicating improved performance robustness. The main reason is that larger warm-up phase lengths provide more informative system behavior and allow AE-LSTM to learn a more accurate representation of the evolving degradation patterns.



(a) Length = 10 (b) Length = 20 (c) Length = 30

(d) Length = 40 (e) Length = 50 (f) Length = 60

Fig. 28. Comparison of different lengths in the warm-up phase.

These results confirm that the warm-up phase length has a fundamental effect on predictive performance, and longer lengths generally lead to more accurate and more stable response prediction at unseen degradation stages. However, extending the warm-up phase delays the deployment of the proposed lifelong update framework, which hinders the timely predictive maintenance in physical systems.